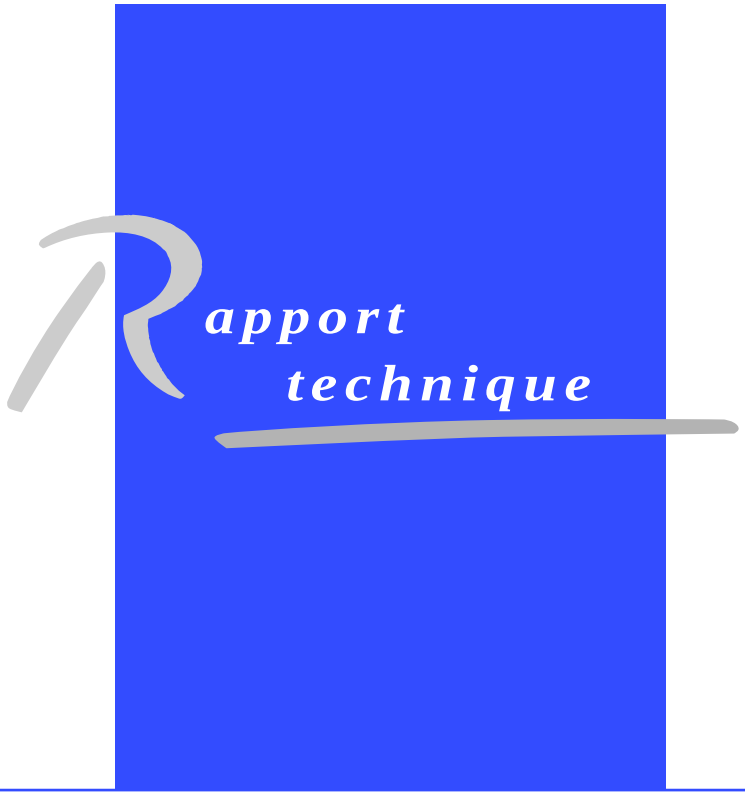
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

CGKAT The User's Reference Manual

Nada Matta - Philippe Martin

N° 0220
May 1998

THEME 3



ISSN 0249-0803

CGKAT, The user's Reference Manual

Nada Matta* - Philippe Martin

Thème 3 : Intelligence artificielle, systèmes cognitifs
et interaction homme-machine

Projet ACACIA

Rapport technique n˚0220 - May 1998 - 116 pages

Abstract: We present in this document a user's manual of the CGKAT tool. CGKAT allows to define knowledge bases with conceptual graphs. These graphs are linked with hypertext links to expertise documents. CGKAT also provides information retrieval from the graph base and from documents. CGKAT offers a generic ontology, defined thanks to a restructuring and combination of some existing ontologies and of the semantically structured dictionary WORDNET.

Key-words: Conceptual Graphs, Information Retrieval, Structured Document, Ontology.

(Résumé : tsvp)

* Email: Nada.Matta@inria.sophia.fr

Unité de recherche INRIA Sophia Antipolis : 2004, route de Lucioles - B.P. 93 - 06902 Sophia Antipolis cedex **(France)**
Téléphone : +33 4 92 38 77 77 - Télécopie : (33) 4 92 38 77 65 - Télex : 970 050 F

CGKAT: Manuel de Référence

Résumé : Ce document présente un manuel d'utilisation du logiciel CGKAT. Ce logiciel permet au cogniticien de construire une base de connaissances représentées dans le formalisme de graphes conceptuels, avec maintenance de liens hypertextes avec les documents d'expertise et aide à la recherche de connaissances dans la base ou d'informations dans les documents. CGKAT offre une ontologie de haut niveau reposant sur une restructuration et une combinaison des ontologies existantes et du dictionnaire sémantiquement structuré WORDNET.

Mots-clé : Graphes conceptuels, Recherche d'Informations, Documents structurés, Ontologies.

Presentation

1 Presentation

CGKAT allows to represent Conceptual Graphs in their context by linking elements (texts and graphs) via hyper-text links. It allows also to search information (conceptual or not) from documents. It offers a general ontology (CGKAT ontology) as help to define conceptual knowledge.

The conceptual graph model [Sowa, 84] is a graphical knowledge representation model, that structures knowledge in two levels, a terminological levels and an assertion level. The terminological level contains the "Support" composed by a concept type lattice and an ordered set of conceptual relations. The assertion level contains graphs defined with respect to a canon. A canon contains the vocabulary of experts to build graphs on an expertise.A conceptual graph is a connected, bipartite, labelled graph. Each graph can be associated to a first order logic formula (Figure 1).

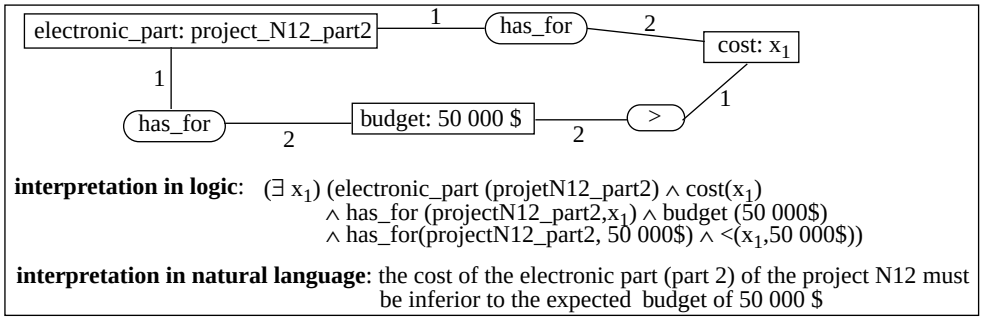


Figure 1 : Example of conceptual graph

Several operations (restriction, simplification, join, projection,...) allow an easy data handling and help to maintain consistency in the conceptual graph base.

1.1 Technical Needs

Sun SparcStation with Sun OS 4.2 or Solaris.

1.1.1 CGKAT Architecture

CGKAT integrates a number of tools (Figure 2) like the Structured Document Editor Thot 3.5 [Quint & Vatton,92], the Conceptual Graphs Platform Cogito 3.2 [Haemmerlé,95] and the Semantic Dictionary WordNet 1.5 [Miller et al,90].

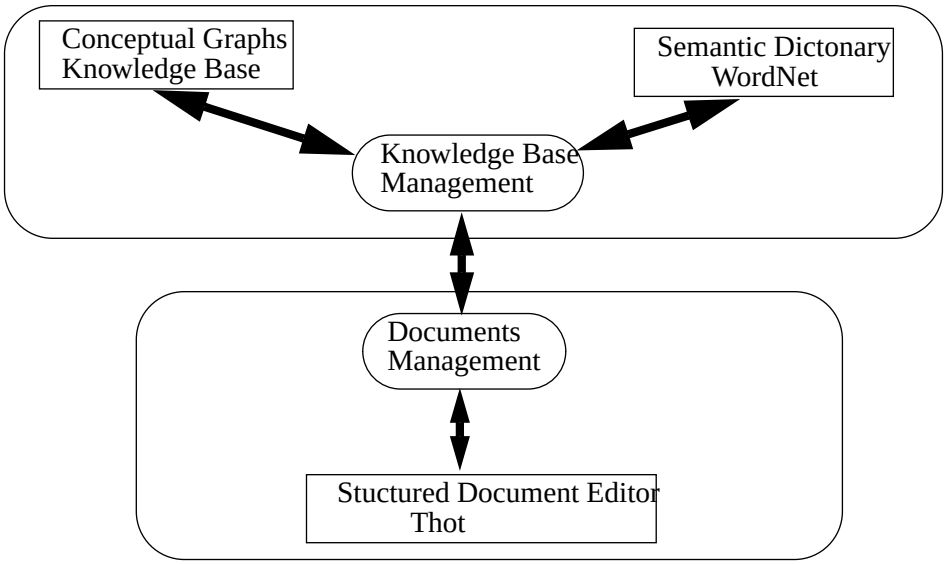


Figure 2 : CGKAT Architecture

2 Organization of the Manual

After a presentation of how to start with CGKAT(Cf. 4), we describe functions that allow to represent Conceptual Graphs as structured documents (Cf. 5) and to search information using Conceptual mechanisms and hyper-text links (Cf. 6). Advanced functionality are then presented(Cf. 7). An example is also described in the Appendix (Cf. 9).

Functions are presented as scenarios of use of CGKAT, with an overview of functions at each step.

Notation:

In this manual we note by:

<name>	A string to be done by the user. For example, <name of file>: a name of a file.

2.1 In case of problems

The current version of CGKAT is a prototype. So, a number of bugs still exist in the system. We indicate the non-function of some functionalities by the expression: Beware and the mark:

Also, Functionalities may not respond well after a number of manipulations in CGKAT. We recommend to exit from CGKAT with saving the session and to launch it.

2.2 Potential users

CGKAT allows as indicated above to represent conceptual graphs. So, user must know the principles of conceptual graphs.

3 Mouse and Keyboard

Select a String	Drag with the left button.
Show a Menu	Press the Right button.
Edit a Link	Double Click with the Left button.
Select an Element (Element Document, Concept, Relation,...)	Click with he Left button on the upper left corner.
Move a Concept or a Relation	Press Ctrl+Left button, Move the Element.

Getting Started

4 Getting Started

In a shell editor do:
cgkat

A panel of menus is shown (Figure 4) with a colour palette (Figure 3) that can be used to change the colour of the editors.

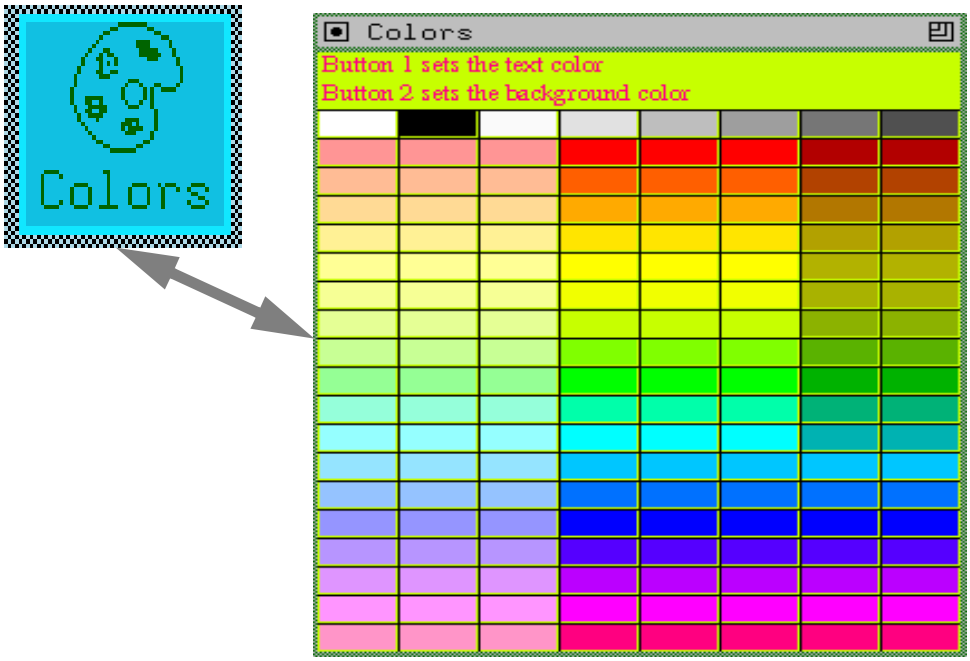


Figure 3 : The color palette.

Documents defined with CGKAT must be in the same directory as the CGKAT program.

The shell editor shows also error and warning messages. It is used also to answer confirmation request, for example to exit from CGKAT, to save files, etc.

CGKAT can be launched with options like:

cgkat -noDefaultEnv	Launch CGKAT without the CGKAT Ontology.
cgkat - e <BCGT-file>	Launch CGKAT and load a file in BCGT format. The BCGT format is linear format used to represent conceptual graphs with Cogito.

4.1 The CGKAT Panel

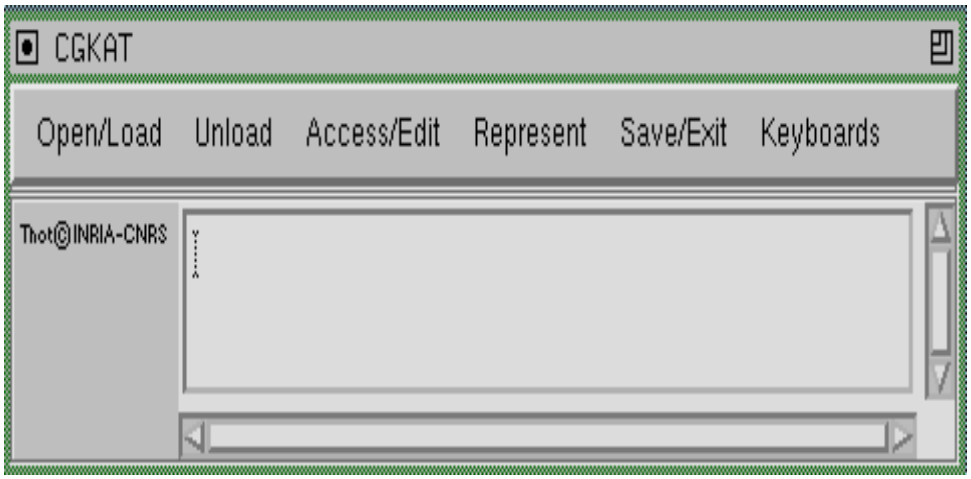


Figure 4 : CGKAT Main Panel

4.2 Open and Load Files

As noted in above, files that can be loaded and saved with CGKAT must be in the same directory as CGKAT program.

Open/Load Create or Load a file, like:

- **Open The Default Document**

  Load the default Document. In the current version, there is no default document.

- **Open Document**

  Open a file. The file format must be the Thot file format: <name>.PIV.
  A menu is shown, one can choose between:

  - Full Document
  - Skeleton

- **Create an Article Document**

  Create a document in Thot format. The file must be saved using the Command "Save as" in the Document Menu. If not, the document is saved in Tmp.PIV.

  A Thot document is structured in elements like Title, Author, Abstract, Sections... (Cf.).

- **Create a CGReprList document**

  Create a list of Conceptual Graphs representations. The document must be saved with "Save as" command. (Cf. 5.2.1).

- **Create a CG document**

  Create a Conceptual Graph document. The document must be saved with "Save as".

- **Create a CG document by copying the selected element**

  Not yet Implemented.

- **Load a BCGT file**

  Load a file in BCGT format. The BCGT format is the linear format used to represent conceptual graphs with Cogito. The name of the file must be included in the CGKAT main shell editor.

- **Load a file which contains a Conceptual Graph in linear format**

  Load a file which contains a Conceptual Graph in linear format. The name of the file must be included in the CGKAT main shell editor. The linear format of a Conceptual graph is as follows: [concept1] -> (relation1) -> [concept2].

- **Load supertypes of uncategorized WordNet Concept**

  Load a supertype of uncategorized WordNet Concept.

- **Start a linear interpreter to access or update the KB**

  Open a CGKAT commands interpreter. These Commands allow to search Knowledge from the Knowledge Base (KB) and to update it. (Cf. 6)

4.3 Access to CGKAT Hierarchies

In conceptual Graphs, hierarchies of concepts and relations can be defined. These hierarchies are organized in type/sub-type.

Access/ Edit

Access and edit CGKAT Hierarchies like:

- **Edit and Use The Concept Type Hierarchy**

  (Cf. 5.4.1)

- **Edit and Use The Relation Type Hierarchy**

  (Cf. 5.4.2)

- **Edit and Use The Instance Type Hierarchy**

  (Cf. 5.4.3)

- **Lexical Search in the Selected Element**

  (Cf. 5.4.4)

4.4 Represent Elements with Conceptual Graphs

Represent Define a Conceptual Graph representation for a selected element from the document. These representations are managed as Hyper-text links (Cf. 5.2). Actions that allow to represent graphs can be:

- **Represent the selected element with a CG (or a type definition)**

  A Conceptual Graph or a type definition. A type definition is a general definition like a lambda abstraction to apply to some types of concepts (Cf. 5.2.3).

- **Represent the selected element with a Single Concept of type Proposition**

  A concept as a subtype of Proposition (Cf. 5.2.4).

- **Represent the selected element with a Single Concept**

  A concept (Cf. 5.2.2).

- **Associate a CGNote to the selected element**

  A note (Cf. 5.2.5).

4.5 Save as a BCGT file and Exit from CGKAT

Save/Exit Allows to:

- **Save concept types and relation types in a BCGT file**

  Save the concepts and relations types defined in a BCGT file format.

- **Save the whole environment in a BCGT file**

- **Exit**

  Exit from CGKAT

4.6 Change Keyboards Characters

Keyboards Write Characters in specifics formats and Symbols like:

- **Math Symbols**

- **Graphics**

- **Latin Alphabet**

- **Greek Alphabet**

To start, we recommend to create an article or to open an existing one (Cf. 5.1.2), then to select a number of elements (words) from the document and to represent them as Single Concepts (Cf. 5.2.2), to select corresponding elements for representing Conceptual Graphs as proposition (Cf. 5.2.4) or type definition (Cf. 5.2.3) using Single Concepts defined and finally to annotate elements using Conceptual Graphs and Single Concepts defined (Cf. 5.2.5). See example in (Cf. 9).

Functionalities

5 Knowledge Representation

CGKAT allows to represent knowledge with conceptual graphs. Links can be defined by the user or by CGKAT between document parts and Conceptual Graphs. A document editor (using THOT [Quint&Vatton,92]) allows to edit documents and to link parts of texts (called Document Element) to Conceptual Graphs that are defined as representations.

5.1 Editing Documents

A document is opened or created using the Menu:

Open Document

The file format must be the Thot file format: <name>.PIV. A menu is shown, where the user can choose between:

Full Document
Skeleton

Create an Article Document

Create a document in Thot format. The file must be saved using the Command "Save as" in the Document Menu. If not, the document is saved in Tmp.PIV.

When you choose to Open a Full Document or to Create an Article, an editor (Figure 5) is shown. A Thot document is structured in elements (called **units** or **document elements**) like: Title, Author, Abstract, Sections... It is organized as a tree of elements:
Document -> Title -> Author
Abstract -> Text
Sections -> Title
Text
....

Conceptual Graphs can be also included as document elements. To do this, you must use the menu **Insert** -> **Others**-> **Conceptual Graphs**... (Cf. 5.1.2).

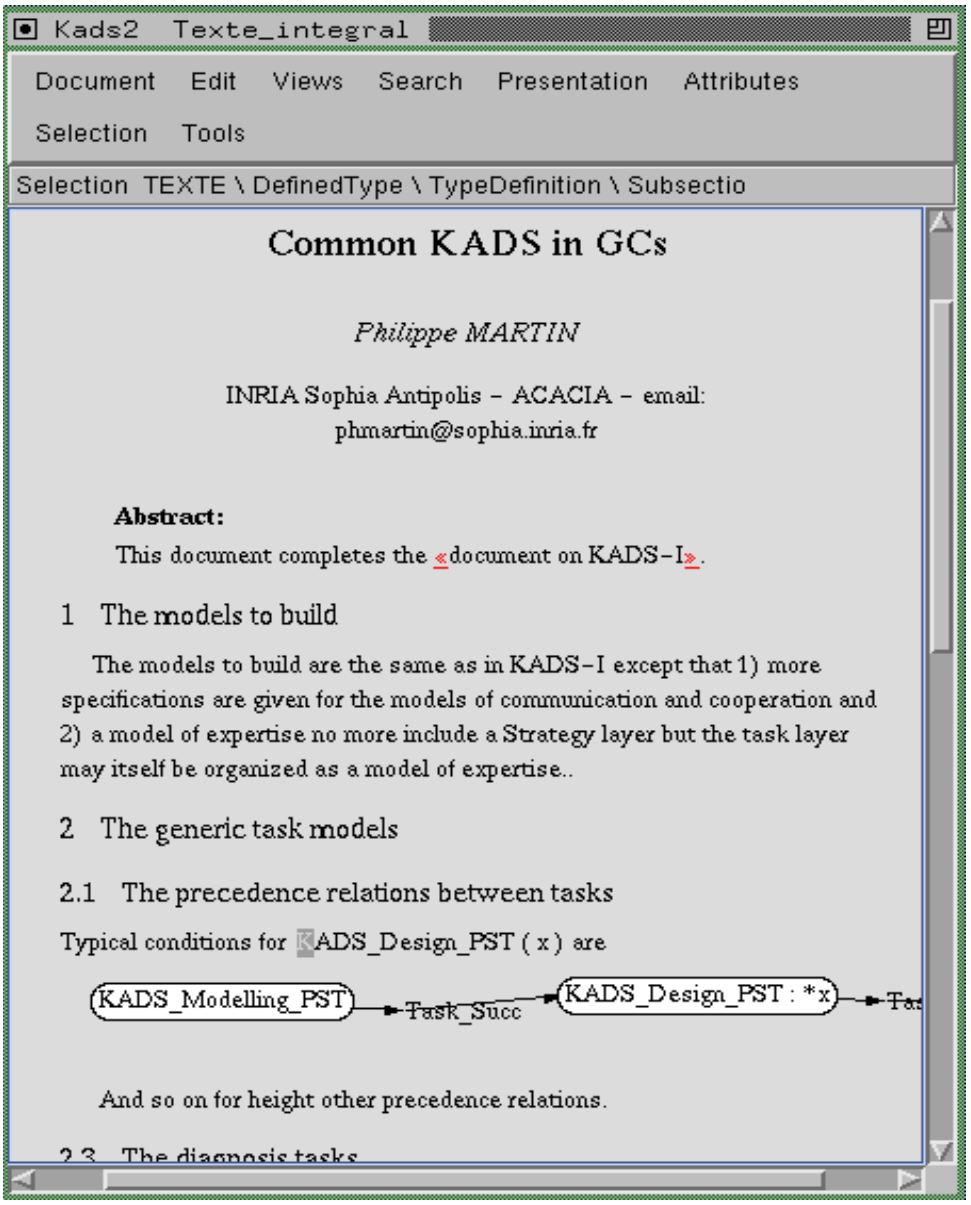


Figure 5 : CGKAT Document Editor

Cursor must be put (by clicking the mouse left button on the place) where a text will be inserted.

5.1.1 General commands about the Document

Document Allows to execute general commands about the document like:

- **Save**

  Save the Document.

- **Save As**

  Save the document with a new name.
  Do not change directory.

- **Autosave**

  Automatic save. User can define the delay between two Automatic save.

- **Pagination**

  Paginate the document.

- **Print**

  Print the document. An asker editor is shown. You can choose printing options, views to be printed, paper format,... Output can be also a Postcript file.

- **Present**

  Presentation of the document.

- **Close this document**

  Close the document and all its related views.

5.1.2 Edit the Document

Edit Allows to execute editing commands like:

- **Insert**

  Insert **mark around, text, units (sections, paragraphs, references, date,...), blocks (units, graphs, picture, other Conceptual Graphs CGs, CG Representations, CG Request,...),...** before or after a unit of the document.
  Elements shown in the **Insert** Menu are contextual.
  The right button of the mouse show also the **Insert** Menu in any place of the cursor on the document.

  Examples: To define a Conceptual Graph after a paragraph,

  - Click with mouse left button in the place where the graph will be inserted.

  - Choose the **Insert** Menu (**Edit** Menu or by pushing the mouse right button).

  - Choose **Block after paragraph**, a menu is shown in which a list of block types is presented (**Paragraph, Picture, Numerated-formula, Figure, Other**),

  - Choose **Other**, a menu is shown (**CGs, CGRep, CGReprList, CGRequests**),

  - Choose **CGs**,

  - Another menu is shown (**SingleConcept, CG, Typedefinition**), then choose the type of the graph from this menu,

  - Finally, define the Conceptual Graph as shown in (Cf. 5.3).

- **Copy**

  Copy a selected element.

- **Paste**

  Paste a copied element.

- **Cut**

  Cut selected element.

- **Include**

  Include is considered as a logical copy of an element. A logical copy can not be modified. Only, the basic element can be modified, which introduce the modifications in all the logical copies of the element.

- **Split-simple-paragraph**

- **Surround**

- **Change Type**

- **Delete**

  Delete selected element.

- **FromXClipboard**

  Paste from X Window.

- **ToXClipboard**

  Copy to X Window.

- **Holophrast**

  Encapsulate a unit and its nature is shown.

  Examples:

  - Having a part of document:

  1. *Kads Models*
  In Kads, several models must be defined.....

  1.1*Expertise Model*

  - Select This element and choose **Holophrast** in the **Edit** Menu, you obtain:

  1<*Section*>

  1.1<*Sub-section*>

5.1.3 Views in the Document

Views Manage different views of the document:

- **Open a View**

  Open a View of the document. This view can be:

  - **Texte-Integral:** The contents of the document.
  - **Table des Matieres:** (This option is in french), the Index of different parts of document. This index is defined automatically (Figure 6).
  - **Notes:** Notes about the document.
  - **Bibliography:** The bibliography of the document.
  - **CGElems:** Conceptual Graphs.
  - **CGNotesOnElems:** Conceptual Graphs represented as annotations of elements of the documents (Figure 10). For more detail, (Cf. 5.2.5).
  - **CGcReprsOnElems:** Conceptual Graphs (Single concepts) defined as representations of elements of the documents (Figure 7). For more detail, (Cf. 5.2.2).
  - **CGReprsOnElems:** Conceptual Graphs defined as representations of elements of the documents (Figure 9). For more detail, (Cf. 5.2.3) and (Cf. 5.2.4).
  - **Index:** Index of marked elements. To obtain this type of index (Figure 8). For more detail, (Cf. 5.1.8).

- **Visibility**

  Allows to modify the visibility of marks inserted around elements. If the visibility is under 5, marks are invisible.

- **Zoom**

  Allows to modify Characters size.

- **List**

  List general information about a given file.

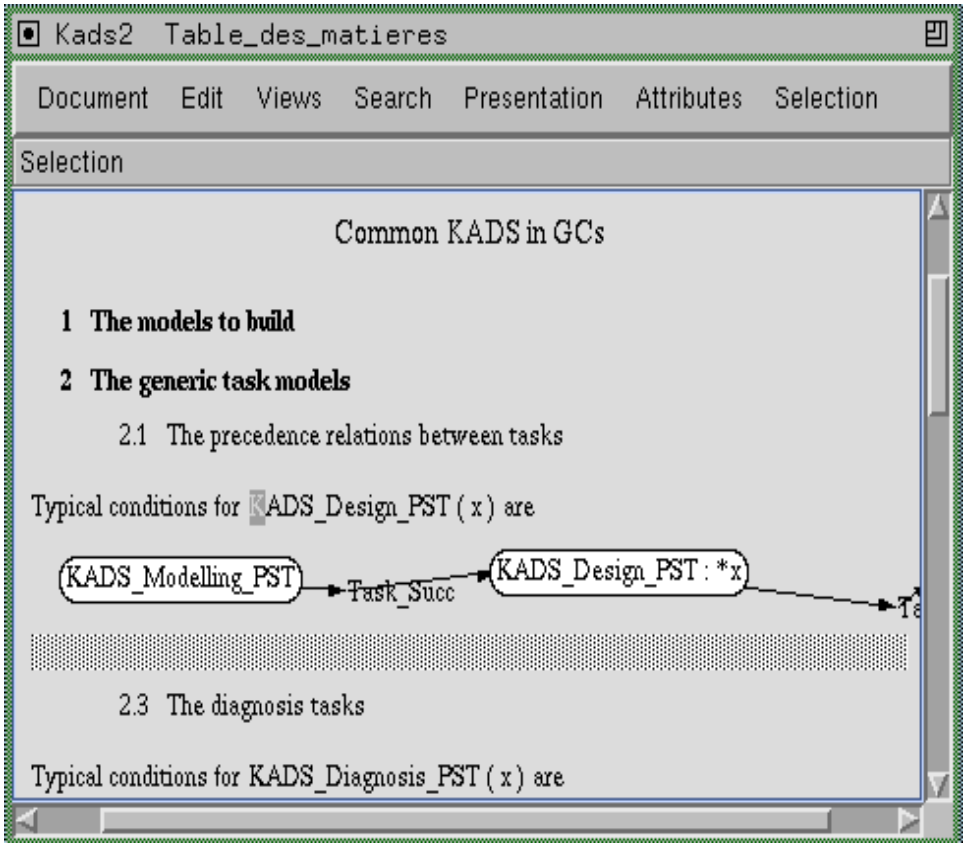


Figure 6 : CGKAT View of Table des Matieres.

5.1.4 Search Elements in the Document

Search Search Elements in the document

- **Search Text and structure**

  Search (and Replace) elements from document or parts of document. Elements to be searched can be: **Text, Graphics, Symbol, picture**,...

- **Search empty elements**

  Search empty elements in the document (For example empty sections).

- **Search empty references**

  Search elements referred to nothing.

5.1.5 Presentation of the document

Presentation Define the format of:

- **Characters**
- **Graphics**
- **Colors**
- **Format** (of the document)
- **Standard Presentation**

  Choose the standard presentation of Characters, Graphics,...

5.1.6 Define Attributes of elements

Attributes Define the attribute of a selected element like:

- **Language**
- **Highlight**
- **Programming**

  Indicates if the element is a programming identifier or a keyword.

- **To_Descriptor**

  Define the description to give to the selected element on the glossary "Descriptor". Elements must be marked around (use **Insert => mark around**). Then, you can link the element to a the descriptor glossary using (**Attributes => To_descriptor**) and you click (with the mouse left button) on the descriptor glossary on the right descriptor element.

- **To_Anything**

  Link the selected element to another element. Elements must be marked around (use **Insert => mark around**). Then, you can link the element to another one by using (**Attributes => To_Anything**) and by clicking (with the mouse left button) on the right element.

- **Foreground Color, Background color**
- **Elem_Depth, Elem_Visibility**

- **Display Help**

  Not yet Implemented in the current version of the system.

- **FreeAttrGLB1**

  Problems appear when using this option. So, we do not recommend to use it.

- **ToCGReprOnElem, ToCGNoteOnElem,ToCGReprsOnElem,ToCGcReprsOnElem,ToCGNotesOnElem.**

  Beware: We do not recommend to use these options. Information defined automatically, when representing conceptual graphs using these option, are not complete. There are another and more properly way to define conceptual graphs with CGKAT (Cf. 5.3).

5.1.7 Selection of elements

Selection Allows to reach:

- ^ **up:** Father of the selected element.
- < **left:** Sibling of the selected element.
- > **right:** Sibling of the selected element.
- **-:** Son of the selected element.

5.1.8 Tools

Tools Present a number of tools like:

- **Spelling Checking**
- **Index to build**

  It allows to define an index of marked terms in the document. This tool is used in CGKAT to define an index of elements represented as concepts. In this case, after defining concepts you choose The **Index to build** tool. A small editor allows to choose **Repagination** option and **Merge the pages** (to gather included concepts and referred ones). More options can also be chosen like: **delete unused descriptors** or **delete used index marks**. For more detail see [Quint&Vatton,92]. Concepts gathered in this index are linked to the text as well as to the representation of the concepts. The index table is accessible via The **view: Index Table**.
  We can also use this tool to build index of a given type of elements. The **Options view** helps to indicate the type of element. In this case, the tool **Index to build** is used as a Thot functionality. Refer to Thot User Guide [Quint et al,94], to build this type of index.

5.2 Represent elements selected from document as Conceptual Graphs

To represent an element of the document as a Conceptual Graph, you must first select the appropriate element (some strings, paragraph,...) and then you choose an appropriate option on the **Represent Menu** of the **CGKAT panel** (Figure 4).

5.2.1 Representation Lists

In a representation list, representations as Conceptual Graphs are gathered. There are three types of representation list: Representation of Single Concepts list (CGcReprList), Representation of CGgraph or Type definition (CGReprList) and Note List (NoteList). A number of Meta-Informations about each representation are included. These Meta-informations invite to define:
User: (The user)
Viewpoint: (The viewPoint under which the representing is defined)
Creation Date: (The creation date of the representation)
Source: From which expert, the source of the representation.
Context of Use: In which context the representation can be used
Comment: Comment about the graph.

Two lines of comments can also be inserted, by clicking (Left button) on the **Comment** field, pressing the (right button) to show the **Insert** Menu and finally choosing: **Free Text Elem1 after Comment** or **Free Text Elem2 after Comment**. A Text entry (visualized by a cursor) is shown after the **Comment** field.

5.2.2 Represent a concept

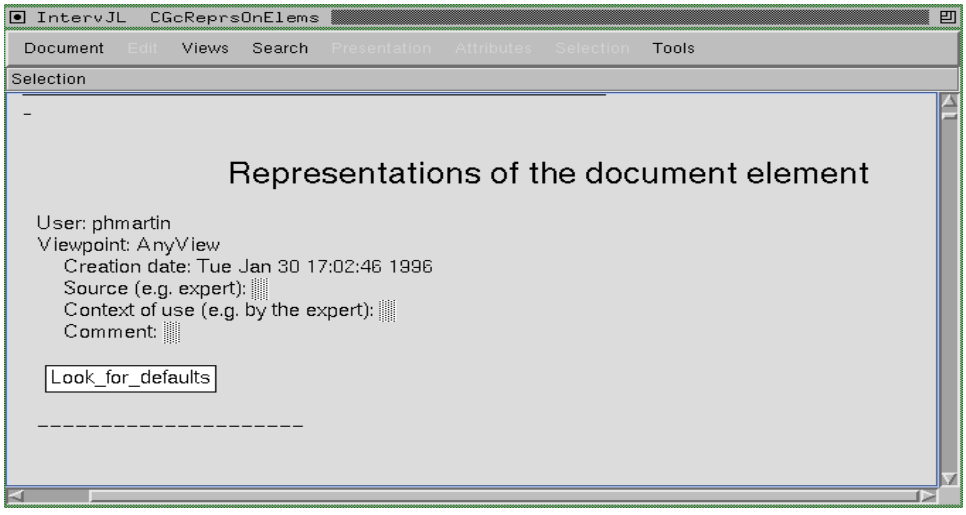


Figure 7 : CGKAT Single Concept Representation List

Represent Create a Conceptual Graph representation related to a selected element from the document. These representations are managed as Hyper-text links. The possible representations are: Single concept, Conceptual Graph, type definition and annotation.

❑ **Represent the selected element with a Single Concept**

Note that the selected element must be a word or a composite word from the document.

✎ Choose The Option: **Represent the selected element with a Single Concept** from **Represent** Menu, in order to represent this element with a concept an editor is shown. It allows to search a concept from the lattice (the CGKAT concept hierarchy) or from WordNet ontology (for more detail, (Cf. 5.4.1)).

✎ Enter an approximate name of the concept (in the appropriate field at right button of the editor) by clicking three times in this field and then you enter the name.

✎ Then, press The button: **Search for concept types including the next string**. A menu is shown that allows to orient the search from the: **Search in the lattice** (the CGKAT Hierarchy), or **Search in WordNet Ontology**.

✎ Then, choose an appropriate concept from the presented list and click the button: **Add a concept, or update the selected concept using the below selected type**. If it is the first concept represented in the document, a small asker is shown, it presents the name of the Representation list. Type **Ok** to confirm your choice.

✎ The appropriate representation list is then shown. You must complete Meta information and you can add a referent to the concept by:

- selecting the concept and choosing the option: **Access to the InstanceHierachy** in the **Access/ Edit** Menu of the CGKAT panel (Cf. 4.1).
- Select the Concept from the list, under which the new instance must be added.
- As in Concept Hierarchy, you must click (left button) three times on the name asker and delete the old name. You enter a new name of the Instance
- select the Option **Add a new instance under it** in the **using the below selected instance** Menu. The Instance is then added in the list of instances shown in the hierarchy.
- Finally, select the corresponding instance from the Instance list and click (left button) on the **Add an individual concept, or update the selected referent, using the below selected string** option. The referent in the concept defined in the representation, is then added. For more detail, (Cf. 5.4.3).

A Link is defined by CGKAT between the selected element and the corresponding concept and vice-versa. (click twice to show the linked element).

After defining all concepts, you can build an index of these concepts by:

✎ choosing the tool **Index to Build** (**Tools** Menu).

✎ A small editor allows to choose **Repagination** option and **Merge the pages** (to gather included concepts and referred ones). More options can also be chosen like: **delete unused descriptors** or **delete used index marks**.

✎ An index table is defined; it is accessible via the View Index Tables (**Views -> Open a View -> Index Tables**) (Figure 8). A reference of a concept is presented as follows in the Index:

<Selected element > (<concept name>, <View>, <User>, <Expert>): <Page number>

Links are defined by CGKAT between the index reference and its corresponding elements respectively (the element from the document, the concept name from the Representation list, the View, User and Expert definition from the Meta Information in the Representation List.

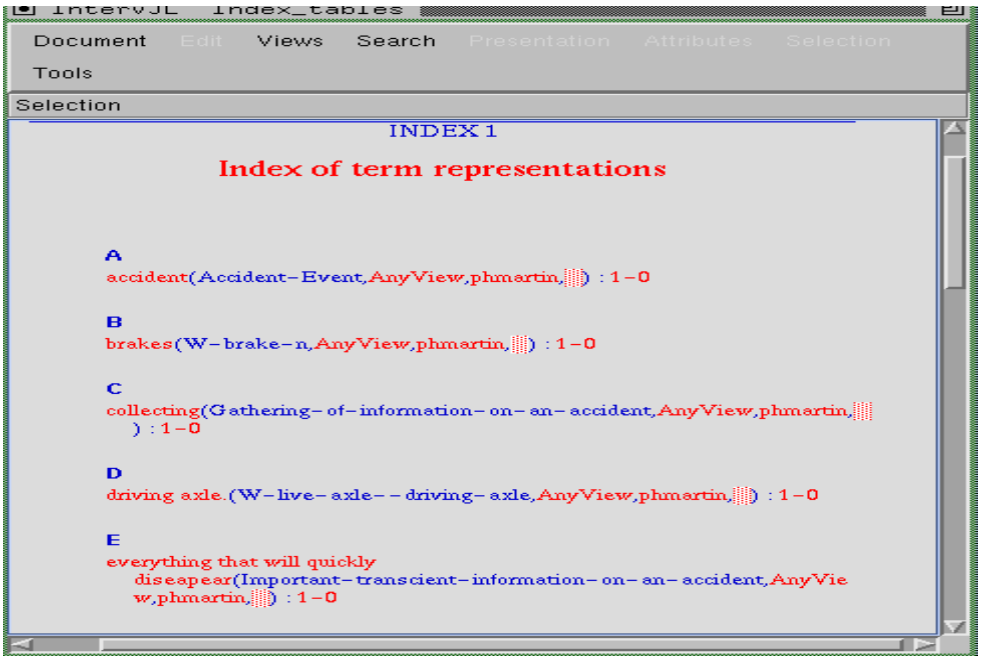


Figure 8 : CGKAT Concept Index

5.2.3 Represent a Conceptual Graph

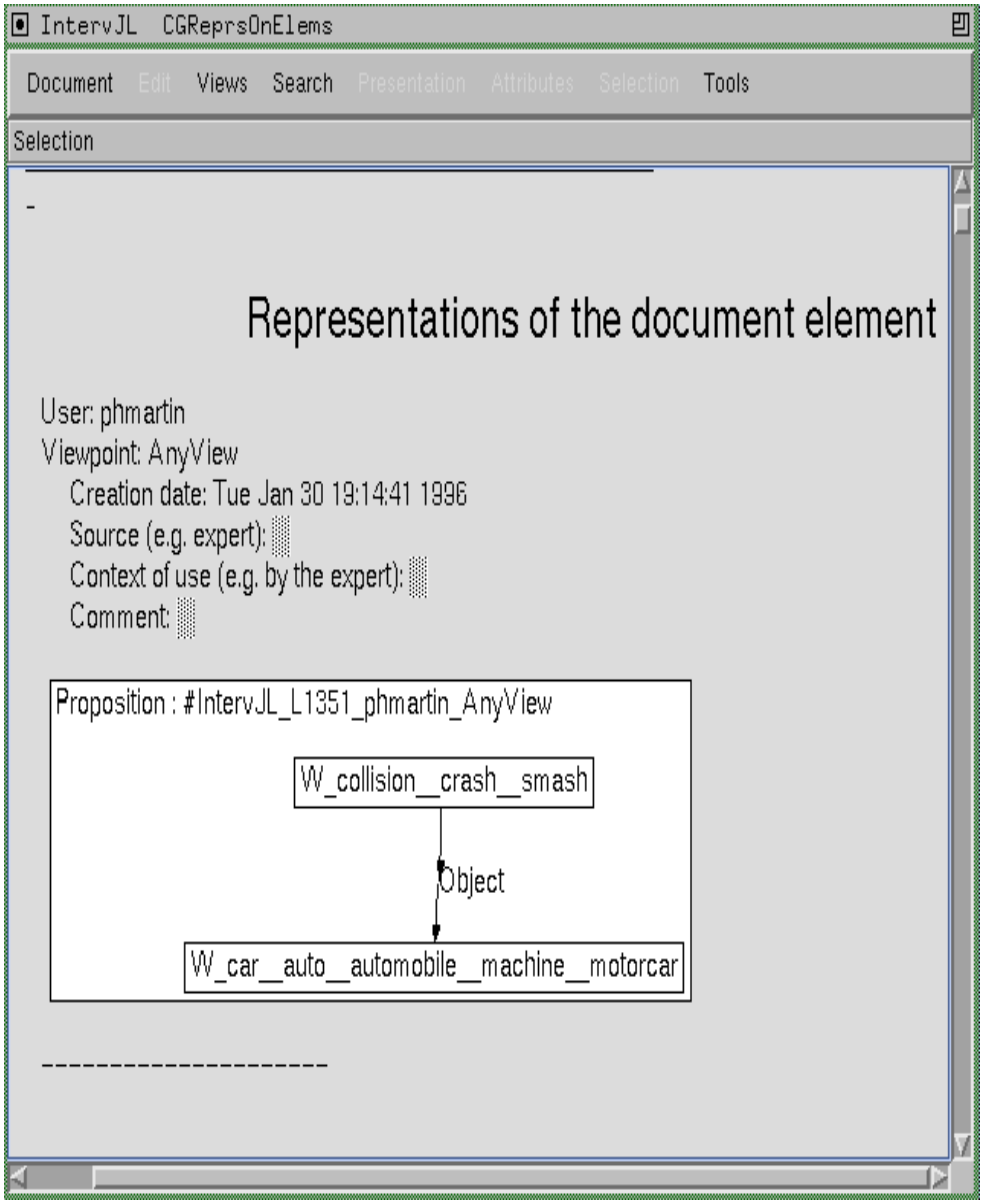


Figure 9 : CGKAT Conceptual Graph Representation List

Represent

- **Represent the selected element with a CG (or a type definition)**

  Allows to represent a Conceptual Graph or a type definition. A type definition is a general definition like a lambda abstraction to apply to some types of concepts (Figure 11).
  If it is the first graph represented in the document, a small asker is shown, it presents the name of the Representation list. Type **Ok** to confirm your choice.
  The appropriate representation list is then shown. You must fulfill Meta information. In the button of Meta Information a small rectangle (like a cursor) show the place to insert a Graph or a type definition. Then:

  ✎ Select this rectangle and then click on the right button to show an appropriate Insert Menu.

  ✎ The Option:**Generated CG after (or before) elem** allows to insert a Graph or a type definition. Another menu is shown.

  ✎ Choose between: **CGgraph** or **Type Definition**.

  ✎ Then, you can add Concepts and Relations to define a graph as indicated in (Part 5.3).

5.2.4 Represent a Conceptual Graph as a Proposition

Beware: insert ConceptInclusion to define Concepts in a Graph. So, Concepts must be already represented with SingleConcept and then, you can include them in a Graph.

A Link is defined between the selected element and the corresponding representation (click twice on the element to show the linked graph).

We recommend to define CGgraph (not Type Definition) as proposition (Cf. 5.2.4). This allows to include the defined graph in another one.

❑ **Represent the selected element with a Single Concept of type Proposition**

Allows to represent a Conceptual Graph noted as a proposition. This notation allows to include the graph in another ones (Figure 9).
If it is the first graph represented in the document, a small asker is shown. It presents the name of the Representation list. Type **Ok** to Confirm your choice.
The appropriate representation list is then shown. You must fulfill Meta Information. A concept called Proposition is shown.

✎ Select the referent of the concept and then click on the right button to show an appropriate **Insert** Menu.

✎ The option: **Generated Referent after individual** allows to insert a Graph. Another menu is then shown.

✎ Choose **CGgraph** in the Menu: **CGgraph**, **Type Definition**.

✎ Then, you can add Concepts and Relations to define a graph as indicated in (Part 5.3).

Beware: insert ConceptInclusion to define Concepts in a graph. So, concepts must be already represented with SingleConcepts and then, you can include them in a graph.

A Link is defined between the selected element and the corresponding representation (click twice on the element to show the linked graph).

5.2.5 Annotate elements

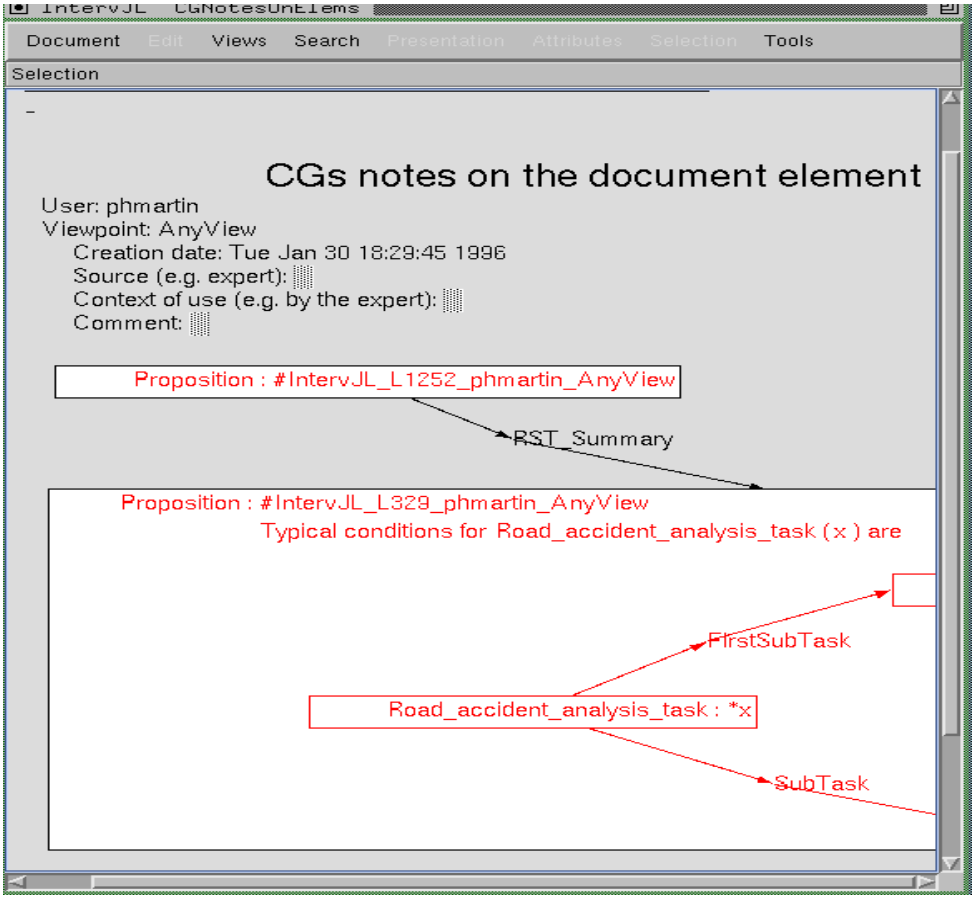


Figure 10 : CGKAT Annotation List

❑ **Associate a CGNote to the selected element**

Allows to annotate elements and representations.
If it is the first note represented in the document, a small asker is shown, it presents the name of the Representation list. Type **Ok** to confirm your choice.
The appropriate representation list is then shown. You must fulfill Meta information. In the button of Meta Information a small rectangle (like a cursor) shows the place where to insert a Graph or a type definition.

✎ Select this rectangle and then click on the right button to show an appropriate **Insert** Menu.

✎ The Option:**Generated CG after (or before) elem** allows to insert a Graph or a type definition.

✎ Another menu allows to show between: **CGgraph** or **Type Definition**. Then, you can add Concepts and Relations to define a graph as indicated in (Part 5.3).

We recommend to include graphs and to link them with another one or with another included concept. To do this, Graphs must be represented as Propositions. Then, these propositions can be included using **ConceptInclusion**.

A Link is defined by CGKAT between the selected element and the corresponding representation (click twice on the element to show the linked graph).

5.3 Define a Conceptual Graph

Define a Conceptual Graph. To do this, the **Insert** Menu, obtained by clicking on the right button of the mouse, allows to Insert a **CGgraph** (Figure 12) or a **Type Definition** (Figure 11). A type definition is a general definition like a lambda abstraction to apply to some types of concepts. In both cases, a Conceptual Graph are defined in the same way. A Conceptual Graph is defined with concepts and binary relations between concepts.
Beware: insert ConceptInclusion to define Concepts in a Graph. So, Concepts must be already represented with SingleConcept and then, you can include them in a Graph.

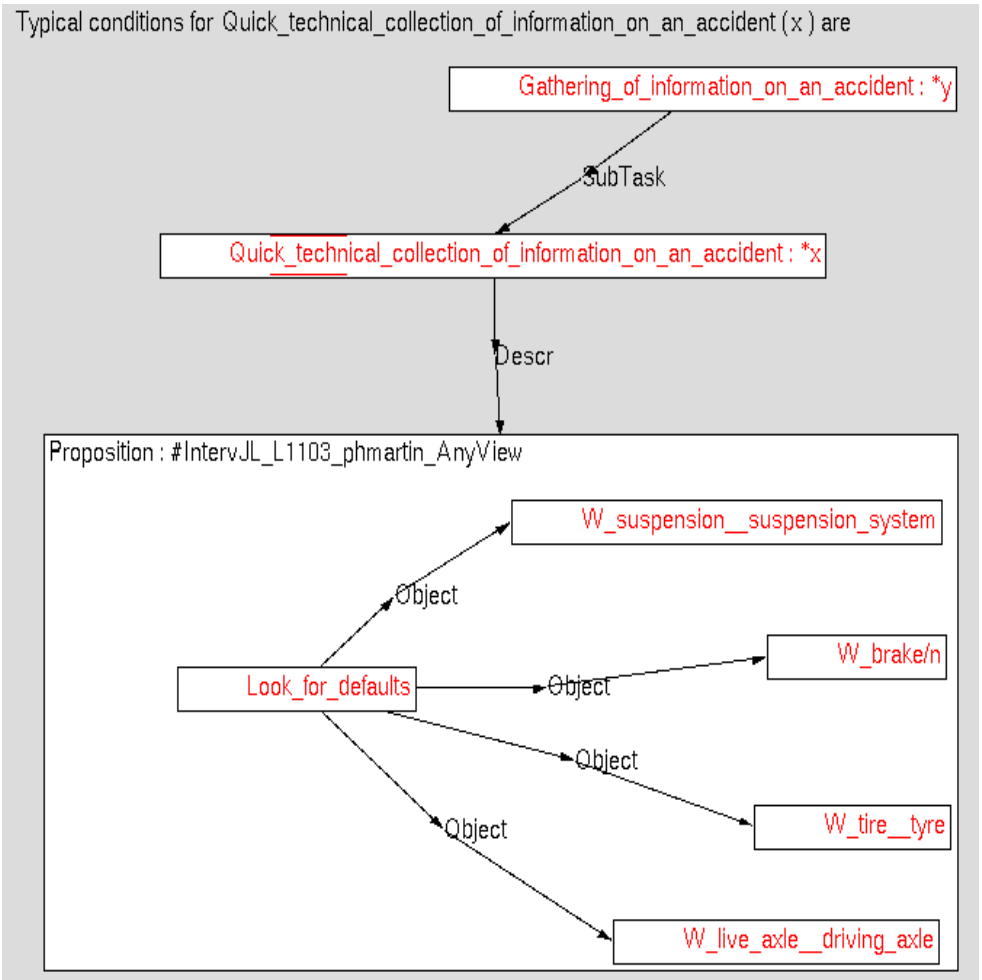


Figure 11 : CGKAT Type definition

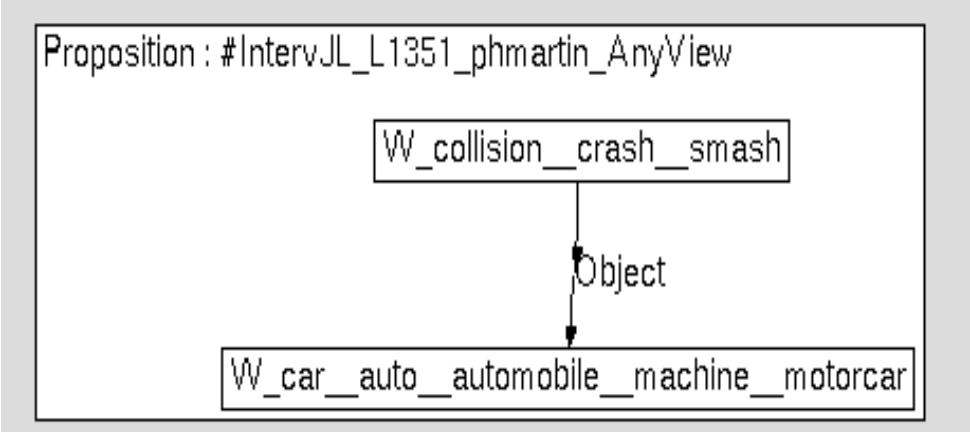


Figure 12 : CGKAT Conceptual Graph as Proposition

5.3.1 Add a concept

CGgraph or Type Definition

Another menu is then shown. It allows to choose the nature of the objects to be inserted in a graph:

- **Concept**

  We do not recommend to use this option to insert concepts in a graph. Concepts so defined are not indicated in the Index of Concepts.

- **ConceptInclusion**

  Allows to include concepts in the graph. When this option is chosen, the user must:

  ✎ click on the concept to include. We recommend to show the corresponding concept (for example, by clicking on the corresponding element in the document) before inserting the ConceptInclusion.

Beware: If no concept has been yet defined in the graph, the first ConceptInclusion does not work. You must insert a concept by choosing **Concept** and not **ConceptInclusion** and then, you include the corresponding concept. Finally, you select the first concept (inserted) and you cut it (Option **Cut** in **Edit** Menu). Then, **ConceptInclusion** works normally.

5.3.2 Add a relation

CGgraph or Type Definition

❑ **ConceptOrRelation**

Allows to define a binary relation between two concepts. These concepts must be already included in the graph. To define a relation, the user:

✎ selects (by clicking with left mouse button on left button of the corner) a concept,

✎ choose the option **ConceptOrRelation** and then the Option **Relation**

✎ and finally, clicks (left button) on the name of the first concept and on the name of the second concept. A draw editor is shown that allows to choose arrow format.

To change the name of the relation defined:

✎ selects the relation

✎ choose the option to the **Access to the Relation Hierarchy** in the **Access/Edit** Menu of the CGKAT panel (Cf. 4.1).

✎ As in Concept Hierarchy, enter an approximate name of the concept (in the appropriate field at right button of the editor) by clicking three times in this field and then you enter the name (for more detail, (Cf. 5.4.2).

The signature of the relation to be chosen must be coherent with corresponding concepts in the graph: The signature of the relation must be compatible to the types of the concepts in the graph. To choose an adequate relation, The Option **List relations with their signature** under the button **Other Kinds of relation type lists**, shows all relations defined with their signature.

- click on the right relation in the relation list
- and add it in the graph by pressing the button **Add a relation, or update the selected relation using the below selected type**.

Another way to choose an appropriate relation is:

- to select the first concept in the graph,
- choose in the Relation Hierarchy editor **List the types of possible relations from a concept of the selected concept type** or **List the type of possible relations from the selected concept** under the button **Other Kinds of relation type lists**,
- Then, select the relation in the graph and choose an adequate relation from the list in the Relation Hierarchy editor
- and finally, press the button **Add a relation from the selected concept, or update the selected relation type**.

You can add a new relation type by adding the name of the relation and its signature (<relation name> (<concept1 name>,<concept2 name>)). For more detail, (Cf. 5.4.2).

5.4 CGKAT Hierarchies

Three types of Hierarchies are managed in CGKAT: a Concept hierarchy, a Relation hierarchy and an Instance list. To access to these hierarchies, you choose the corresponding option from the **Access/Edit** Menu in the CGKAT Panel (Figure 4).

5.4.1 Concept Hierarchy

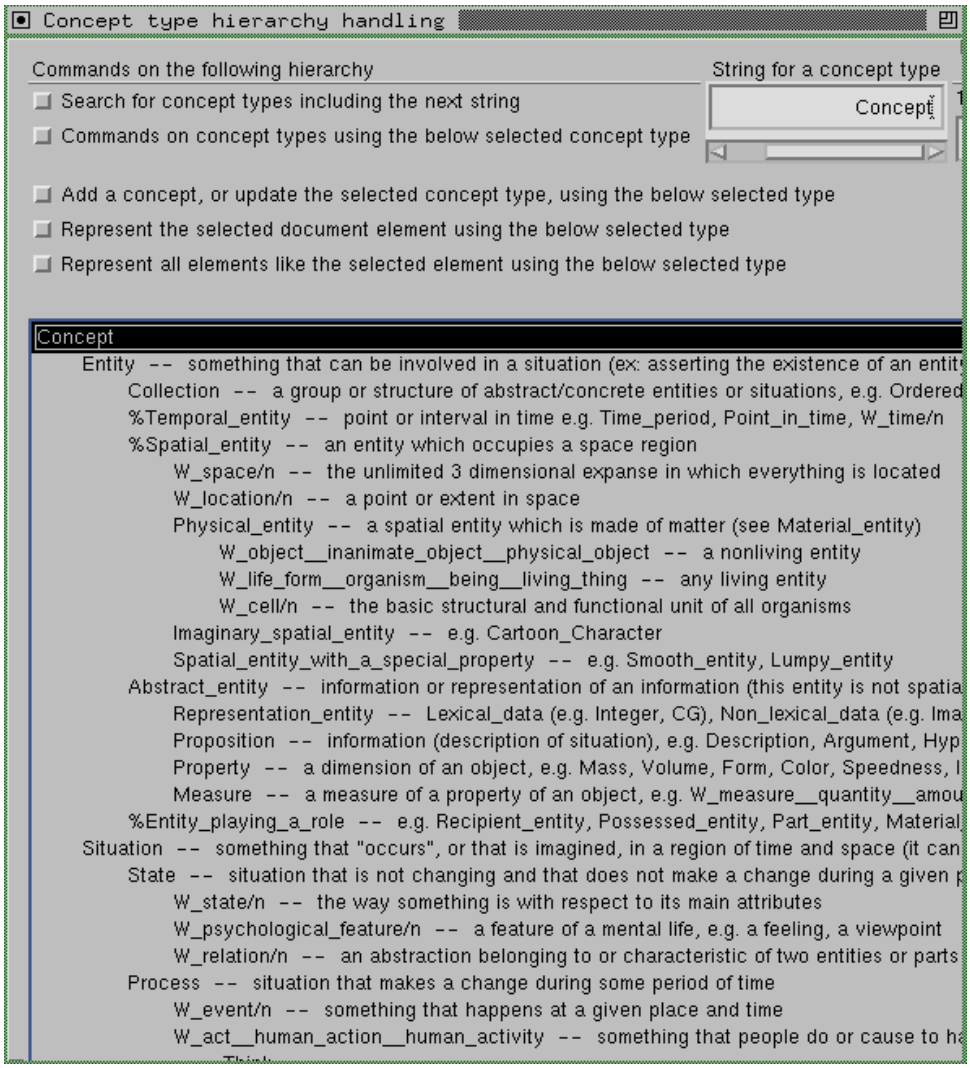


Figure 13 : CGKAT Concept Hierarchy editor

Access/ Edit

❑ **Edit and Use the Concept type Hierarchy**

An editor shows the hierarchy of concepts as an indented list (Figure 13). A number of buttons and menus allow to search a concept type from the hierarchy and to manage it. At the bottom of the hierarchy window, there are three commands. To access to these commands, you must move up the window. These commands allow to: **Search** (Not yet Implemented), **Help** (Not yet Implemented), **Reload** (Reload the hierarchy window) and **Cancel** (Close the hierarchy window).

➪ **Search for concept types including the next string**

To search a concept from the lattice (the CGKAT concept hierarchy) or from WordNet ontology, you must:

✎ Enter an approximate name of the concept (in the appropriate String field at top right of the editor) by clicking three times in this field and then you enter the name.

✎ Then, press the button: **Search for concept types including the next string**. A menu is shown:

✎ choose between **Search in the lattice** (the CGKAT Hierarchy), or **Search in WordNet Ontology**. A list of corresponding concept types are presented.

✎ Then, choose an appropriate concept from the list.

➪ Command on Concept type using the below selected concept type

Allows to manage the concept hierarchy. A menu is shown. You can choose options like:

➥ Add a new concept under it

✎ The concept to add is the one already selected from the list.

✎ Then, select another concept under which the first selected concept must be added

✎ and choose the option: **Add a new concept under it** in the **Command on Concept type using the below selected concept type** Menu. The concept already selected is so added under the last selected one.

➥ Alias it with the following concept type name
Not yet implemented.

➥ Add a comment to it

✎ The concept must be already selected from the list.

✎ Then, choose the option: **Add a comment to it** in the **Command on Concept type using the below selected concept type** Menu.

✎ Type the comment in the string field.

➥ Add it as a child of an existing concept type

✎ The concept to add is the one already selected from the list.

✎ Then, select another concept under which the first selected concept must be added

✎ and choose the option: **Add it as a child of an existing concept type** in the **Command on Concept type using the below selected concept type** Menu. The concept already selected is so added as a child of the last selected one.

➥ **Move it and its descendants in another parent**

✎ The concept to move is the one selected already from the list.

✎ Then, select another concept under which the first selected concept must be added

✎ and choose the option: **Move it and its descendants in another parent** in the **Command on Concept type using the below selected concept type** Menu. The concept already selected is then moved as a child of the last selected one.

➥ **Keep this temporary included WordNet Concept type and its supertypes in the lattice**

✎ The WordNet Concept must be first selected from the list.

✎ Then, choose the option: **Keep this temporary included WordNet Concept type and its supertypes in the lattice** in the **Command on Concept type using the below selected concept type** Menu. The concept is then added in the lattice and its supertypes.

➥ **Delete it, its descendants will belong to its father**

✎ The Concept to delete must be already selected from the list.

✎ Then, choose the option: **Delete it, its descendants will belong to its father** in the **Command on Concept type using the below selected concept type** Menu. The concept is then deleted and its descendants are added as children of the father of the deleted concept.

➥ **Delete it and its descendants**

✎ The Concept to delete must be first selected from the list.

✎ Then, choose the option: **Delete it and its descendants** in the **Command on Concept type using the below selected concept type** Menu. The concept and its descendants are then deleted.

➥ **List its descendants which have many parents**

✎ The corresponding concept must be already selected from the list.

✎ Then, you choose the option: **List its descendants which have many parents** in the **Command on Concept type using the below selected concept type** Menu. A list of the selected concepts descendants which have many parents is presented in the CGKAT shell window.

➥ **Say if it is a subtype of the next concept**

✎ The corresponding concept must be already selected from the list.

✎ Then, choose the option: **Say if it is a subtype of the next concept** in the **Command on Concept type using the below selected concept type** Menu. The answer is shown in the CGKAT shell window.

➥ **Display its definition (necessary, sufficient or type definition)**

✎ The corresponding concept must be already selected from the list.

✎ Then, choose the option: **Display its definition (necessary, sufficient or type definition)** in the **Command on Concept type using the below selected concept type** Menu. The definition of the concept is then shown in the CGKAT shell window.

➥ **Save the whole hierarchy as a BCGT file**
Save the hierarchy in a BCGT format.

➪ **Add a concept, or update the selected concept using the below selected type**

✎ Select first the concept to modify from the corresponding conceptual graph (in a representation list or in the document).

✎ Then, choose a corresponding concept from the concept hierarchy: search an appropriate one and select it from the list of concepts shown (Cf. 5.4.1).

✎ Finally, click (left button) on **Add a concept, or update the selected concept using the below selected type**, the concept in the conceptual graph is then updated.

➪ **Represent the selected document element using the selected type**

Beware: we do not recommend to use this command. There is some bugs in this functionality.

✎ Select a word or a set of words from the document.

✎ Then, choose the appropriate concept from the hierarchy (search an appropriate one and select it from the list of concepts shown (Cf. 3.2.2)

✎ and click on **Represent the selected document element using the selected type**, a CGcRepresentation list are shown. It represents the chosen document element by a single concept corresponding to the concept selected from the hierarchy.

➪ **Represent all elements like the selected document element using the selected type**

✎ Select a word or a set of words from the document.

✎ Then, choose the appropriate concept from the hierarchy (search an appropriate one

✎ Select it from the list of concepts shown (Cf. 3.2.2)

✎ Then, click on **Represent all element like the selected document element using the selected type**, a CGcRepresentation list is shown. It represents all elements, that have the same strings as the chosen element, by a single concept corresponding to the concept selected from the hierarchy.

5.4.2 Relation Hierarchy

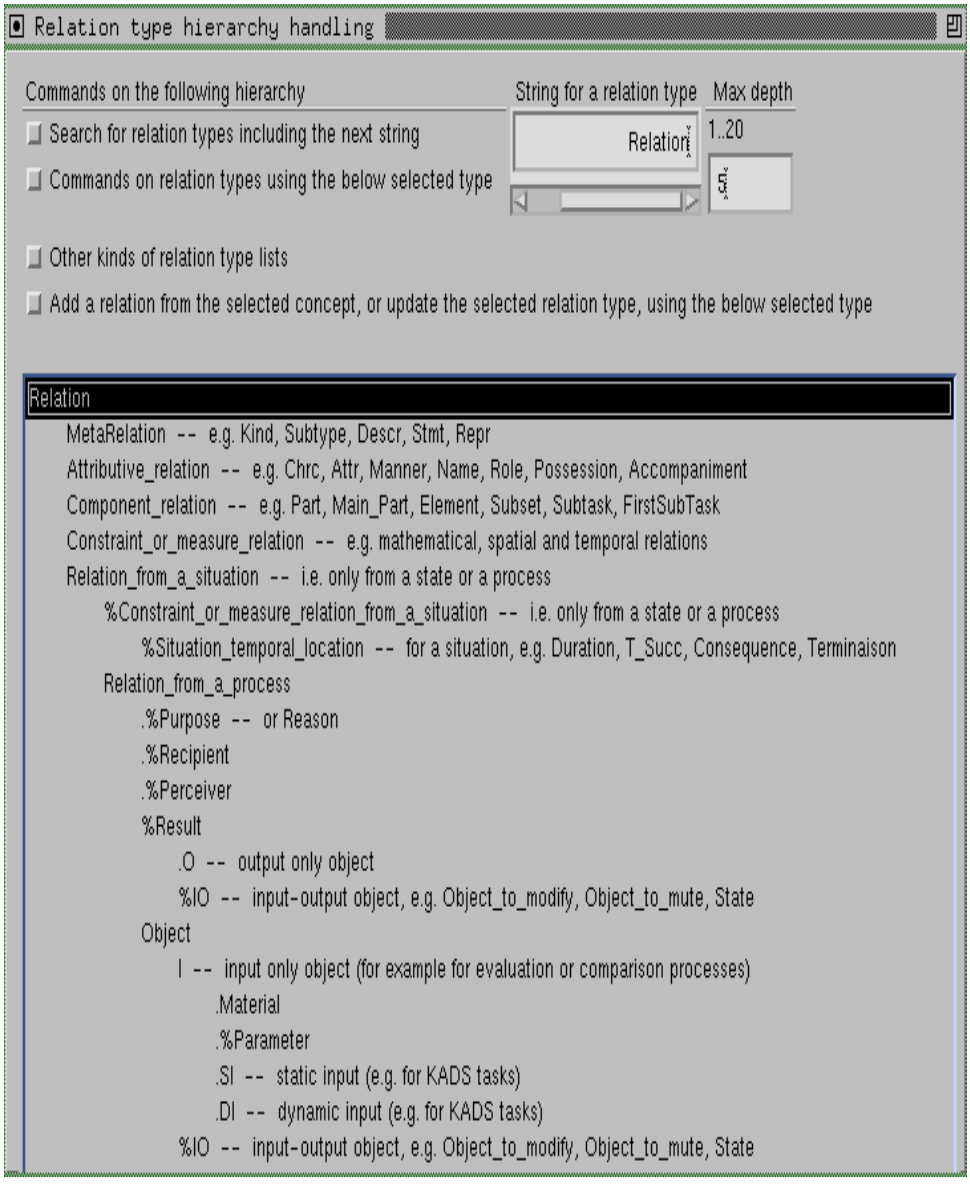


Figure 14 : CGKAT Relation Hierarchy editor

Access/ Edit

❑ **Edit and Use the Relation type Hierarchy**

An editor shows the hierarchy of relations as an indented list (Figure 14). A number of buttons and menus allow to search a relation type from the hierarchy and manage it. At the button of the hierarchy window, there are three commands. To access to these commands, you must move up the window. These commands allow to: **Search** (Not yet Implemented), **Help** (Not yet Implemented), **Reload** (Reload the hierarchy window) and **Cancel** (Close the hierarchy window).

➪ **Search for relation types including the next string**

To search a relation from the lattice, you must:

✎ Enter an approximate name of the relation (in the appropriate String field at top right of the editor) by clicking three times in this field and then you enter the name.

✎ Then, press the button: **Search for relation types including the next string**. A list of corresponding relation types are presented,

✎ Finally, choose an appropriate relation by clicking on (left button).

➪ **Command on Relation type using the below selected type**

Allows to manage the relation hierarchy. A menu is shown. You can choose options like:

➥ **Add a new instance under it**

✎ Choose the relation type from the list, under which the new relation must be added.

✎ Then, type in the string field the new relation to add with its signature respecting the following format: <Relation name> (<concept1 name>,<concept2 name>)

✎ choose the option: **Add a new relation under it** in the **Command on Relation type using the below selected type** Menu. The new relation is then added under the selected one.

Beware: the signature of the relation to add must be coherent with the signature of the relation under which the new relation must be added. If not, an error message is shown in the CGKAT shell.

➥ **Alias it with the following relation type name**
Not yet implemented.

➥ **Add a comment to it**
The relation must be already selected from the list.

✎ Then, you choose the option: **Add a comment to it** in the **Command on Relation type using the below selected type** Menu.

✎ Type the comment in the String field.

➥ **Add it as a child of an existing relation type**
The relation to add is the one already selected from the list.

✎ Select another relation under which the first selected relation must be added

✎ and choose the option: **Add it as a child of an existing relation type** in the **Command on Relation type using the below selected type** Menu. The relation already selected is then added as a child of the last selected one.

Beware: the signature of the relation to add must be coherent with the one under which the relation must be added. If not, an error message is shown in the CGKAT shell.

➥ **Move it and its descendants in another parent**
The relation to move is the one already selected from the list.

✎ Select another relation under which the first selected relation must be added

✎ and choose the option: **Move it and its descendants in another parent** in the **Command on Relation type using the below selected type** Menu. The relation already selected is then moved as a child of the last selected one.

➥ **Delete it, its descendants will belong to its father**
The relation to delete must be first selected from the list. Then, choose the option: **Delete it, its descendants will belong to its father** in the **Command on Relation type using the below selected type** Menu. The relation is then deleted and its descendants are added as children of the father of the deleted relation.

➥ **Delete it and its descendants**
Beware: This command does not work.

➥ **List its descendants which have many parents**
The corresponding relation must be first selected from the list. Then, choose the option: **List its descendants which have many parents** in the **Command on Relation type using the below selected type** Menu. A list of the descendants of the selected relation which have many parents is presented in the CGKAT shell window.

➥ **Say if it is a subtype of the next relation**
The corresponding relation must be first selected from the list. Then, choose the option: **Say if it is a subtype of the next relation** in the **Command on Relation type using the below selected type** Menu. The answer is shown in the CGKAT shell window.

➥ **Display its definition**
The corresponding relation must be first selected from the list. Then, choose the option: **Display its definition** in the **Command on Relation type using the below selected type** Menu. The definition of the relation is then shown in the CGKAT shell window.

➥ **Save the whole hierarchy as a BCGT file**
Save the hierarchy in a BCGT format.

⇨ Other Kinds of Relation type lists

This option allows to display the list of relation types in different format.

➥ List relation types without signature
Allows to show in the list only names of relation types.

➥ List relation types with their signature
Allows to show in the list names of relation types and their signature. This list helps to respect the same format and the coherence of signatures when you want to add a new relation type or to move a relation under another one.

➥ List the type of the possible relations from the selected concept type
The corresponding concept type must be first selected from the Concept Hierarchy. Then, choose the option: **List the type of the possible relations from the selected concept type** in the **Other Kinds of Relation type lists** Menu. A list of possible relation types, having the selected concept type in their signature, is shown.

➥ List the type of the possible relations from the selected concept
The corresponding concept must be first selected from a conceptual graph (in a representation list or in the document). Then, choose the option: **List the type of the possible relations from the selected concept** in the **Other Kinds of Relation type lists** Menu. A list of possible relation types, having the selected concept in their signature, is shown.

➪ **Add a relation from the selected concept, or update the selected relation type using the below selected type**

- Select first the relation to modify from the corresponding conceptual graph (in a representation list or in the document).
- Then, choose a corresponding relation from the relation hierarchy: search an appropriate one and select it from the list of relation types shown (Cf. 5.2.3).
- Finally, click (left button) on **Add a relation from the selected concept, or update the selected relation using the below selected type**, the corresponding relation in the conceptual graph is so updated.

5.4.3 Instance Hierarchy

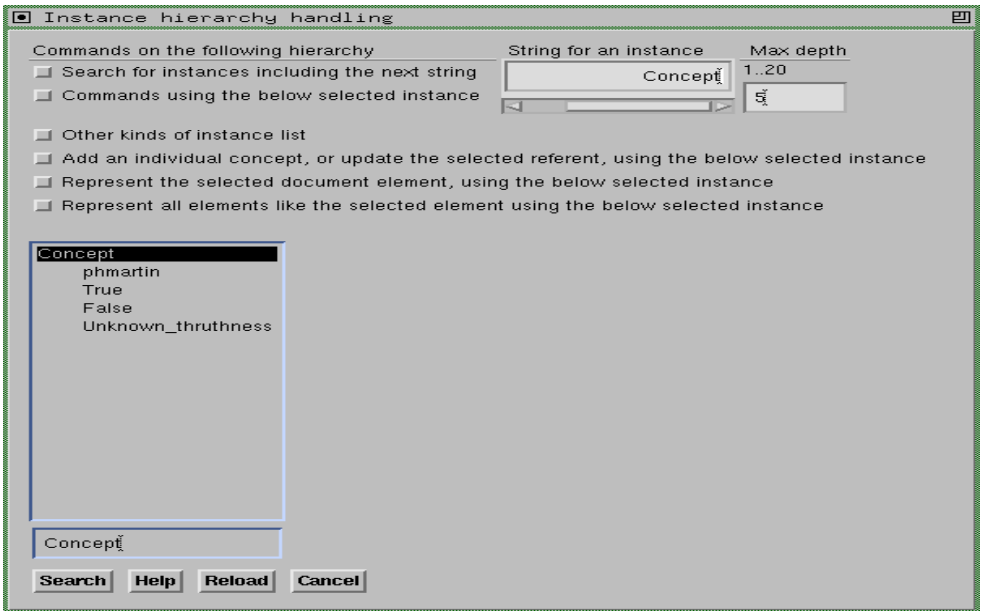


Figure 15 : CGKAT Instance Hierarchy editor

Access/ Edit

❑ **Edit and Use the Instance Hierarchy**

An editor shows the hierarchy of instances as an indented list (Figure 15). A number of buttons and menus allow to search an instance from the hierarchy and manage it. At the button of the hierarchy window, there are three commands. These commands allow to: **Search** (Not yet Implemented), **Help** (Not yet Implemented), **Reload** (Reload the hierarchy window) and **Cancel** (Close the hierarchy window).

➪ **Search for instance including the next string**

To search an instance from the lattice,

✎ enter an approximate name of the instance (in the appropriate String field at top right of the editor) by clicking three times in this field and then you can enter the name.

✎ Then, press The button: **Search for instance including the next string**. A list of corresponding instances are presented with their relative Concepts (as their father).

✎ Finally, the appropriate instance can be chosen by clicking on (left button).

➪ **Command using the below selected instance**

Allows to manage the instance hierarchy. A menu is shown. You can choose options like:

➥ **Add a new instance under it**

✎ Choose the Concept from the list, under which the new instance must be added.

✎ Then, type in the string field the new instance to add and choose the option: **Add a new instance under it** in the **Command using the below selected instance** Menu. The new instance is then added under the selected concept.

➥ **Alias it with the following relation type name**
Not yet implemented.

➥ **Add a comment to it**
Not yet implemented.

➥ **Add it as a child of an existing instance**
Beware**:** This command does not work. There is some bugs in this functionality.

➥ **Move it and its descendants in another parent**
Beware: This command Does not work. There is some bugs in this functionality.

➥ **Delete it, its descendants will belong to its father**
Beware: This command Does not work. There is some bugs in this functionality.

➥ **Delete it and its descendants**
The instance to delete must be first selected from the list. Then, choose the option: **Delete it and its descendants will** in the **Command using the below selected instance** Menu. The instance is then deleted.

➥ **Say if it is an instance of the next concept type (the new concept type must be in the 'String for instance' text form)**
The corresponding instance must be first selected from the list.

✎ Then, type the name of the concept in the String field.

✎ Finally, choose the option: **Say if it is an instance of the next concept type** in the **Command using the below selected instance** Menu. The answer is shown in the CGKAT shell window.

➥ **Save the whole hierarchy as a BCGT file**
Save the hierarchy in a BCGT format.

➪ **Other Kinds of Instance lists**

This option allows to show the list of relation types in different format.

➥ **List the instances of the concept type selected in the 'Concept type Hierarchy handing' menu**
The corresponding concept type must be first selected from the Concept Hierarchy. Then, choose the option: **List the instances of the concept type selected in the 'Concept type Hierarchy handing' menu** in the **Other Kinds of Instance lists** Menu. A list of instances, related to the selected concept type, is shown.

➥ **List the instances of the selected concept or concept type in a document**
The corresponding concept must be first selected from a conceptual graph (in a representation list or in the document). Then, choose the option: **List the instances of the selected concept or concept type in a document** in the **Other Kinds of Instance lists** Menu. A list of instances, related to the selected concept type, is shown.

➪ **Add an individual concept, or update the selected referent, using the below selected string**

✎ Select first the referent to modify from the concept in the corresponding conceptual graph (in a representation list or in the document).

✎ Then, choose a corresponding instance from the Instance hierarchy (search an appropriate one and select it from the list of instances shown (Cf. 5.4.3).

✎ Finally, click (left button) on **Add an individual concept, or update the selected referent, using the below selected string**, the corresponding referent in the conceptual graph is then updated.

➪ **Represent the selected document element using the below selected instance**

✎ Select a word or a set of words from the document.

✎ Choose the appropriate instance from the hierarchy: search an appropriate one and select it from the list of instances shown (Cf. 3.2.2)

✎ Then, click on **Represent the selected document element using the below selected instance**. A representation in the CGcRepresentation list is created. In this representation, a single concept, corresponding to the concept considered as father of the selected instance, is created. The concept represented has as referent the chosen instance. You must click twice (left button) on the element on the document to show the created representation.

➪ **Represent all elements like the selected document element using the below selected instance**

✎ Select a word or a set of words from the document.

✎ Choose the appropriate instance from the hierarchy (search an appropriate one and select it from the list of instances shown (Cf. 3.2.2)

✎ Then, click on **Represent all elements like the selected document element using the below selected instance,** a CGcRepresentation list is then shown. In this representation, a single concept, corresponding to the concept considered as father of the selected instance, is created. The concept represented has as referent the chosen instance. All elements, in the document, which have the same string as the chosen element are linked by CGKAT to this concept.

5.4.4 A list of terms

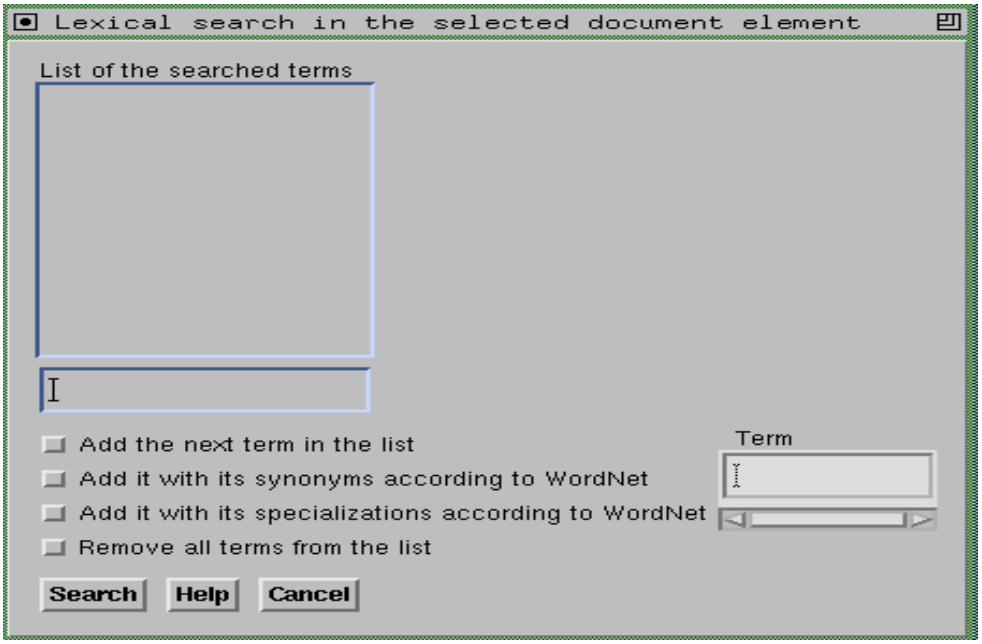


Figure 16 : CGKAT List of Terms editor

Access/ Edit

❏ **Lexical Search in the selected element**

An editor shows a list of terms and a number of commands to create and manage the list of terms (Figure 16). At the button of the window, there are three commands. These commands allow to: **Search** (Not yet Implemented), **Help** (Not yet Implemented), **Reload** (Reload the hierarchy window) and **Cancel** (Close the hierarchy window).

➪ **Add the next term in the list**

✎ Enter the term to add in the appropriate field (Term field) by clicking three times in this field and then you enter the name.

✎ Then, press The button: **Add the next term in the list.** The term is then added in the list.

Access/ Edit

➪ **Add it with its synonyms according to WordNet**

✎ Enter the term to add in the appropriate field (Term field) by clicking three times in this field and then you enter the name.

✎ Then, press The button: **Add it with its synonyms according to WordNet.** The term is so added in the list with its synonyms found in the WordNet Ontology.

➪ **Add it with its specializations according to WordNet**

✎ Enter the term to add in the appropriate field (Term field) by clicking three times in this field and then you enter the name.

✎ Then, press The button: **Add it with its specializations according to WordNet.** The term is then added in the list with its specialization found in the WordNet Ontology.

➪ **Remove all terms from the list**

Remove all terms from the list.

6 Information Search

CGKAT provides a number of functions to search information from documents. This search can be either a simple navigation using hypertext links (by clicking twice on the linked elements) or via request commands that performs conceptual search from the Conceptual Graphs Base generated by Cogito. Documents, in which search is performed, must be opened.

To realize this conceptual search, a number of commands can be used in the interpreter editor or in a shell editor (in this case, you must type ck <command>, i.g. ck spec [cat]). To access to the interpreter editor, use **Open Load** Menu in the CGKAT panel (Figure 4) and choose **Start a linear interpreter to access or update the KB** option. An editor is shown in which commands can be typed. answers are shown after each commands.

Beware: To enter a command, you must select it (use the first button) and push the **Esc** button or the up arrow (as to go to the father). To type another line in a command, push <ctrl return>.

Conceptual graphs must be in the linear format in commands. You can also refer a Conceptual graph by its name. In this case, type # before the name. you can use % before a name of a concept when you are not sure that it is the exact name of the concept.

Beware: only one % can be used in a command.

6.1 Search Commands

These commands allow to do a particular search of information from the Conceptual Graph Base.

➪ **spec** [T1]

Search *subtypes* or i*nstances* of the *type* T1. The search is done until depth 3 of sons.

➪ **spec** #CG1

Search Conceptual Graphs that are specializations of the CG1 Conceptual Graph. The parameter of this command can be also a conceptual graph in a linear format instead the name of the graph. i.g. spec [cat] -> (on) -> [Mat].

➪ **gene** [T1]

Search *supertypes* of *types* (or *types* of *instances*) which include the term T1.

➪ **gene** #CG1

Search Conceptual Graphs that are generalizations of the CG1 Conceptual Graph. Parameter of this command can be also a conceptual graph in a linear format instead the name of the graph. i.g. spec [cat] -> (on) -> [Mat].

➪ **type of** [I1]

Search the type of instance I1.

➪ **def of** [T1]

Search *Type Definition graphs* used for defining T1.

Options can be used in these searches like:

➥ **in** [T1]: Indicates a concept type that is included in Conceptual Graphs.

➥ **with** {attribute1:value1; attribute2:value2;...}: Indicates a number of attribute of Conceptual Graphs**.**

You can indicate the context of search and the form of presentation of informations. You can indicate the type of context and the format of answers before to ask a number of search commands.

For example:

on CGs (to indicate that the search must be done only among the Conceptual Graphs)
use linear (to indicate that you want the answer in linear format)
spec [cat] -> (on) -> [%Mat]
gene [%car]

on lambdas (to indicate that the search must be done only among type definitions. This command dismiss the effect of the command: on CGs).

Commands to specify the context of search and the answer format can be:

- **Definition of context of search**

  For each command you can indicate the context of the search using the command **on** like:

  - **on CGs**

    Search is done only among Conceptual Graphs.

  - **on lambdas**

    Search is done only among Type Definitions.

  - **on all**

    Search is done among all types of document elements.

- **Definition of format of answer**

  For each command you can indicate the format of answer by using the command **use**:

  - **use CGs**

    The answer will be presented in the form of Conceptual Graphs. If found Conceptual Graphs are already defined in the document and not generated for the answer, these graphs are presented graphically. In the other case, graphs are presented in the linear format.

  - **use Repr**

    The answer will be presented as representation of Document elements.

  - **use Annot**

    The answer will be presented as annotation of Document elements.

➪ **use Assoc**

Not yet implemented. Answer will be presented by using association of hypertext links.

➪ **use Repr&Annot**

The answer will be presented as representation and annotation of Document elements.

➪ **use Repr&CGs**

The answer will be presented as representation of Document elements and as graphs.

➪ **use Annot&CGs**

The answer will be presented as annotation of Document elements and as graphs.

➪ **use Repr&Annot&CGs**

The answer will be presented as representation, as annotation of Document elements and as graphs.

➪ **use linear**

Graphs in the answer will be presented in linear format.

You can also omit some type of informations from the answer as:

➪ **no meta**

No meta information will be presented in the answer.

➪ **meta**

The answer can contain meta informations. By default, meta information is done in the answer. This command is used to dismiss the command **no meta**.

➪ **context**

All graphs referring to the graph answer are presented.

➪ **no context**

Graphs referring to the graph answer are not presented. This command is used to dismiss the command **context**.

7 Advanced Functions

A number of commands can be typed in the interpreter editor or in a shell editor (in this case, you must type ck <command>, i.g. ck [cat]). To access to the interpreter editor, use **Open Load** Menu in the CGKAT panel (Figure 4) and choose **Start a linear interpreter to access or update the KB** option. An editor is shown in which commands can be typed. answers are shown after each command. Documents, in which commands are performed, must be opened.

Beware: To enter a command: you must select it (use the first button) and push the **Esc** button or the up arrow (as to go to the father). To type another line in a command, push <ctrl return>.

Conceptual graphs must be in the linear format in commands. You can also refer a Conceptual graph by its name. In this case, type # before the name. You can use % before a name of a concept when you are not sure that it is the exact name of the concept.

Beware: Only one % can be typed in a command.

7.1 Test command

Such commands are used to test information in the base.

➪ #CG1 < #CG2

Test if the graph CG1 specializes the graph CG2.

➪ [T1] < [T2]

Test if T1 is a subtype of T2.

➪ **proj** #CG1 #CG2

Project the graph CG1 on the graph CG2.

➪ **treeproj** #CG1 #CG2

Project the graph CG1 on the graph CG2. (CG1 must have a tree structure).

7.2 Assertion commands

Such commands are used to assert information or to declare a new one.

➪ #CG1

(re)asserts the graph CG1 and displays its content.

➪ **!** #CG1

Assert the graph CG1.

➪ **!** [T1]

Declare the concept type T1.

➪ **!** [I1] **:** [T1]

Declare the instance I1 of the concept T1.

➪ **!** R1 (T1,...Tn)

Declare the relation R1 between concepts T1,...Tn.

➪ [T1] < [T2] ....[Tn]

Declare that the concepts T2,... Tn are subtypes of T1.

➪ [T1] > [T2] ....[Tn]

Declare that the concept T1 is a subtype of concepts T2,... Tn.

➪ **NCS for** [T1] (x) **are** #CG1

Give a definition of the concept type T1 with necessary and sufficient conditions. The conditions is defined by the graph CG1.

➪ **NCS for** [R1] (x1,...xn) **are** #CG1

Give a definition of the relation R1 with necessary and sufficient conditions. The conditions is defined by the graph CG1.

➪ **NC for** [T1] (x) **are** #CG1

Give a definition of the concept type T1 with necessary conditions. The conditions is defined by the graph CG1.

➪ **SC for** [T1] (x) **are** #CG1

Give a definition of the concept type T1 with sufficient conditions. The conditions is defined by the graph CG1.

➪ **TC for** [T1] (x) **are** #CG1

Give a typical definition of the concept type T1. The definition is done by the graph CG1.

➪ **name** #CG1 <new-name>

Rename the graph CG1.

➪ **copy** #CG1 #CG2

Copy CG1 under another name: CG2.

7.3 Graphs generation

Such commands allow to generate graphs by joining it with another ones, based on given concepts. These concepts are considered as roots from which the joining is realized. Joining can be:

➪ **ijoin on** [T1] [T2] #CG1 {<#CG-result-name>}

internal join (joining sub-graphs) on the same graph based on concepts T1 and T2.

➪ **join on** [T1] [T2] #CG1 # CG2{<#CG-result-name>}

external join between two graphs based on concepts T1 and T2.

➪ **isojoin on** [T1] [T2] #CG1 #CG2{<#CG-result-name>}

isojoin on two graphs (format for concepts: type:[:referent]). The joining can also be in this case, based on the specialization of the root concepts.

➪ **maxjoin** {<-n #CG-result-name>} #CG1 #CG2 {#CG3 ...#CGn}

maximal isojoin between CG1...CGn based on all concepts.

7.4 Deletion commands

These commands allow to delete concepts and relations from the base.

- **delCT** [T1] {[T2]...[Tn]}

  Delete concepts T1,...Tn from the base if no graph uses them.

- **delRT** (R1) {(R2)...(Rn)}

  Delete relations R1,...Rn from the base if no graph uses them.

- **delI** [I1] {[I2]...[In]}

  Delete Instances I1,...In from the base if no graph or concept uses them.

- **delCG #**CG1 {#CG2...#CGn}

  Delete graphs CG1,...CGn.

- **delLambda #**L1 {#L2...#Ln}

  Delete type definition graphs L1,...Ln. The defined type becomes atomic but is not deleted.

- **delCGs**

  Delete all graphs.

- **closeDoc** <Doc1-name> {<Doc2-name>...<Docn-name>}

  Close opened documents Doc1...Doc2.

7.5 Load command

Allows to load documents.

- **openDoc** <Doc1-name> {<Doc2-name>...<Docn-name>}

  Open Thot documents Doc1...Doc2. Graphs defined in each document are loaded in the Base.

➪ **load** <BCGCT-file1-name> {<BCGCT-file2-name>...<BCGCT-filen-name>}

Load BCGCT files.

➪ **loadCG** #CG1 <linear-graphs-file1-name>

Load the file "linear-graphs-file1-name" in which CG1 is in linear format.

7.6 Save command

Allows to save documents.

➪ **save env** <BCGCT-file1-name>

Save the environment in a BCGCT file.

➪ **save support** <BCGCT-file1-name>

Save the support in a BCGCT file.

➪ **save** #CG1 <BCGCT-file1-name>

Save the graph CG1 in a BCGCT file. If no file name is given, the graph is saved in .G_$CG1.

➪ **saveLCG** #CG1 <linear-graphs-file1-name>

Save the graph CG1 in linear format (without meta-information).

7.7 Trace commands

Allows to give a trace of a session.

➪ **listCGs**

List graphs defined in the session.

➪ **trace**

Give a trace of the session.

➪ **no trace**

Dismiss the command **trace**.

7.8 Other commands

➪ **script** <script-file1-name>

Execute commands from the <script-file1-name>. Commands can be gathered in a script file (use a simple editor, for example emacs, to define the script file), and executed in CGKAT using the command **script**. Each command in the script file must be ended by a "." and a return.

➪ **:** comments

Do not execute the arguments. This command can be useful in script files.

➪ **sh** command

Execute a UNIX command.

➪ **linearForm** #CG1 {#CG2...#CGn}

Give the linear format of the graphs CG1...CGn.

➪ **display** #CG1 {#CG2...#CGn}

Display the graphs CG1...CGn in linear format or as document elements. This command can be useful in script files.

➪ **setenv** variable {value}

Set environment variables. You can use also **set** instead **setenv**.

8 Interaction

CGKAT can use conceptual graphs defined by other systems. These graphs must be saved in a BCGCT files. So, you can load these files using CGKAT and work on them normally.

A function defined in our team, generates conceptual graphs from a CommonKADS representation of an expertise model. These conceptual graphs are saved in a BCGCT file. So, CGKAT can use them.

Another tool MultiKat that compare conceptual graphs (developed in our team) can be linked also through BCGCT files to CGKAT.

9 Appendix: Example of Use

In this appendix, we present an example of use of CGKAT. We suggest (1) to write documents or to translate them in Thot format, (2) to define conceptual graphs by defining concepts and then graphs, and (3) to search information from both documents and the conceptual graphs base, defined.

9.1 Edit Documents

To create a new document use (in CGKAT Main Panel):
Open/Load -> **Create an Article Document**
Then, you edit your document (Figure 17), by writing texts and inserting sections. Use up, down, left and right arrows to access to father, sons and siblings document elements.

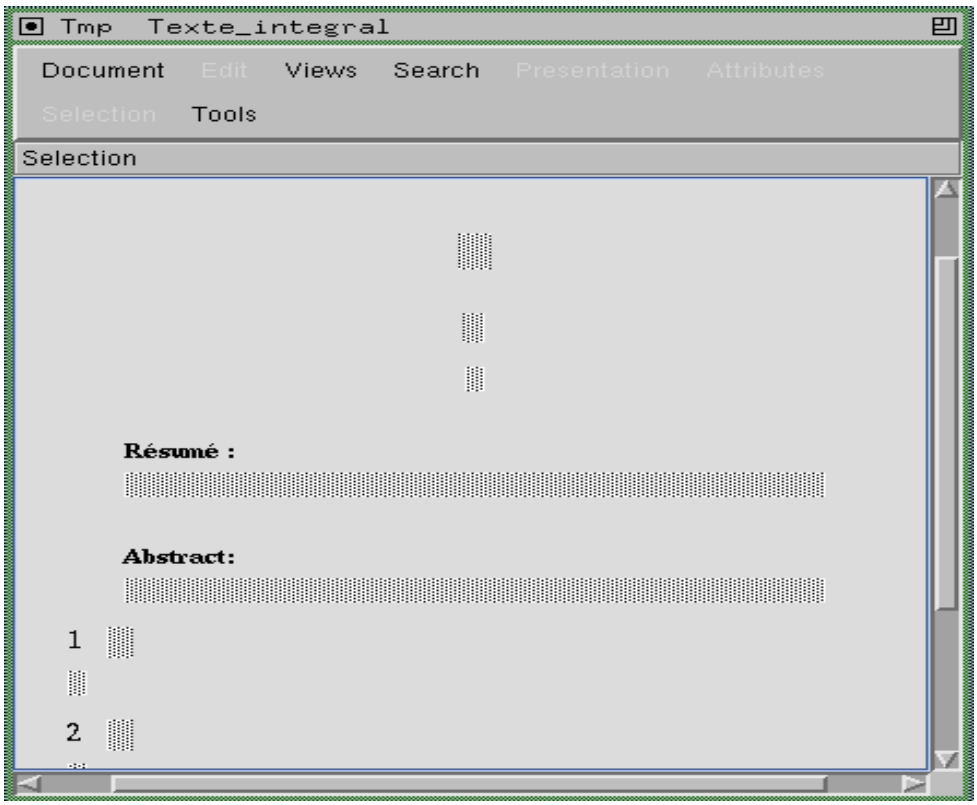


Figure 17 : A new document

You can also open a Thot file (*.PIV) (Figure 18) using (in CGKAT Main Panel):
Open/Load -> **Open Document**

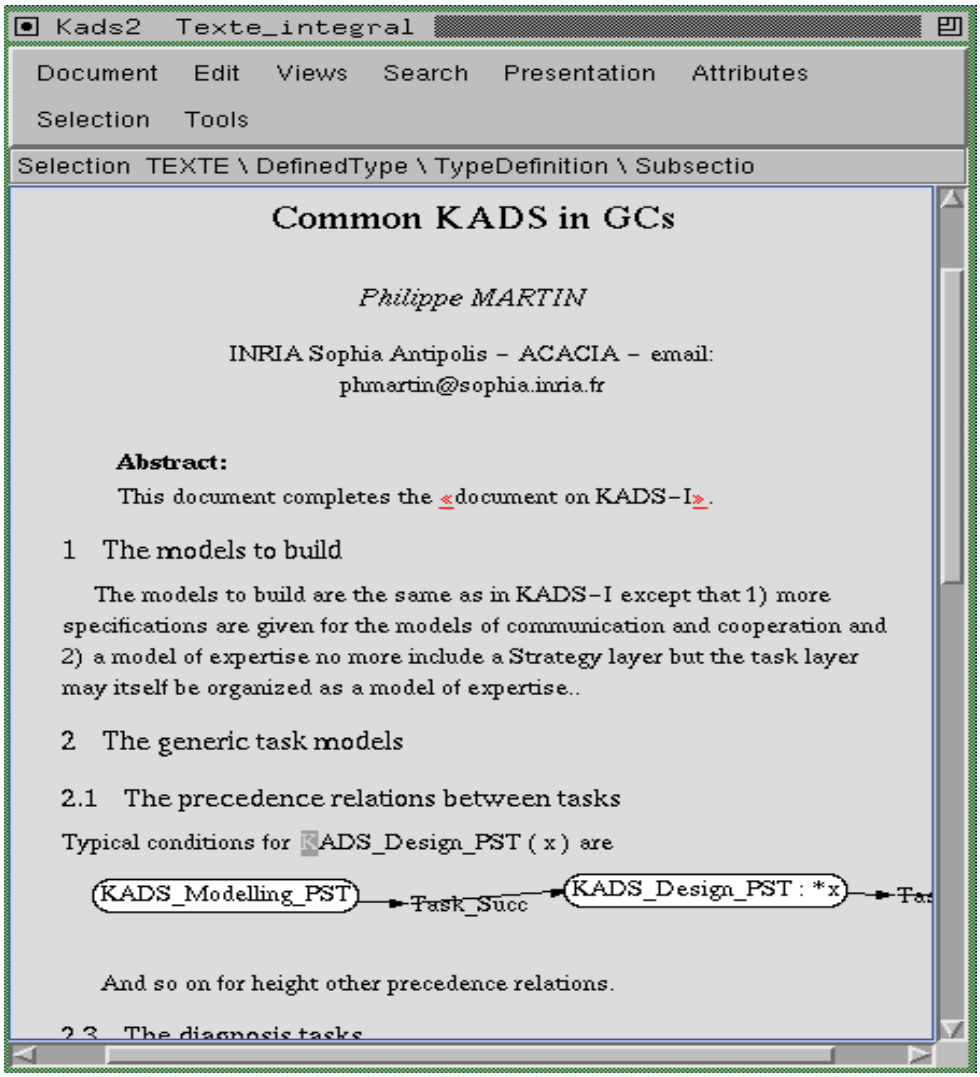


Figure 18 : Open an existing Document

9.2 Define Conceptual Graphs

Concepts to include in graphs must be first represented as Single Concepts.

9.2.1 Represent Single Concepts

- select a corresponding document element (a word or a set of words from the document)
- and then choose in the CGKAT main panel the Menu (Figure 4): **Represent** -> **Represent the selected element with a Single Concept**

An editor is shown. It allows to search a concept from the lattice (the CGKAT concept hierarchy) or from WordNet ontology (for more detail, (Cf. 3.4.1).

- Enter an approximate name of the concept (in the appropriate field at right bottom of the editor) by clicking three times in this field and then you enter the name.
- Then, press The button: Search for concept types including the next string. and choose an option: **Search in the lattice** (the CGKAT Hierarchy), or **Search in WordNet Ontology**.
- Then, choose an appropriate concept from the presented list.
- Finally, click the button: **Add a concept, or update the selected concept using the below selected type** (Figure 19).

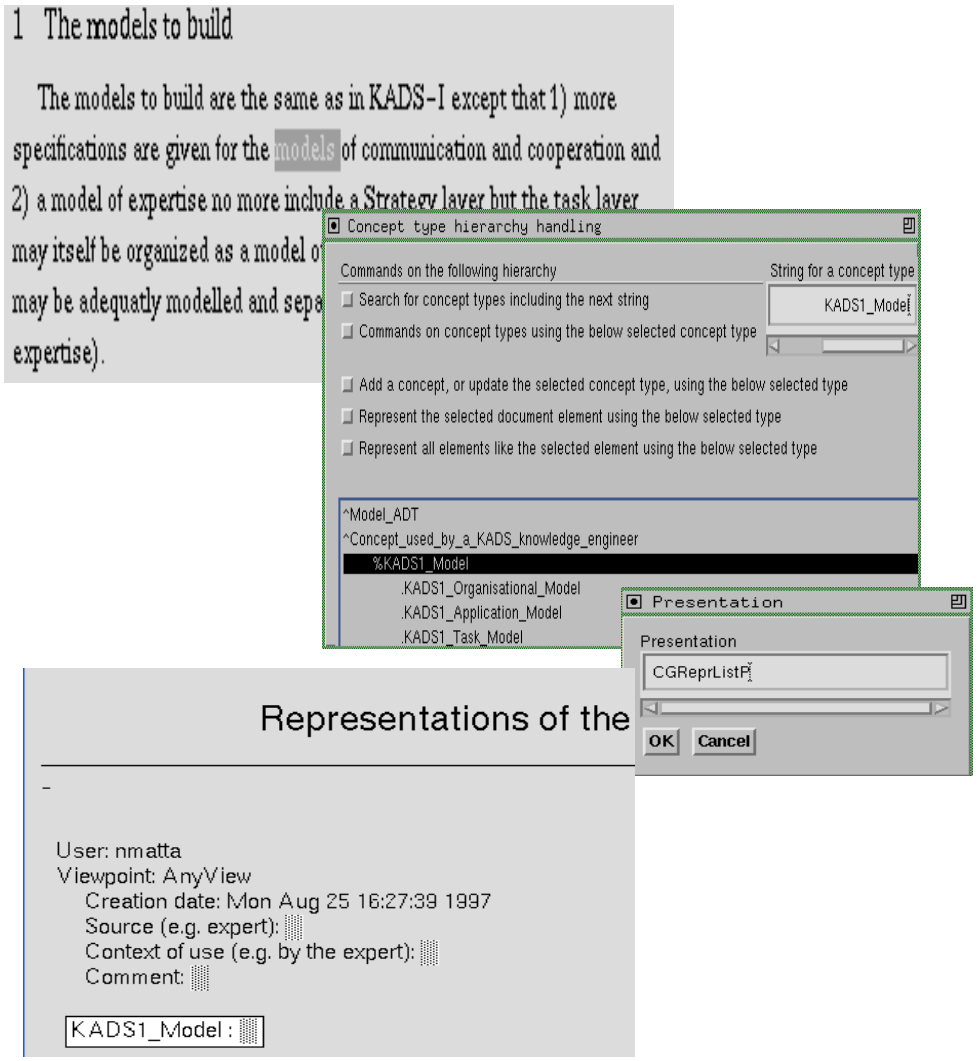


Figure 19 : Represent a single concept

9.2.2 Represent Conceptual Graphs (CGgraphs)

To insert conceptual graphs directly in the documents:

- select a paragraph,
- then, click on the right button of the mouse to show the Insert menu.

Insert Menu

→ **Block after (or before) paragraph**

→ **Other**

→ **CGs**

Or to represent a document element (paragraph or a number of sentences) as a graph:

- select first the document element to represent,
- then, choose in the CGKAT main panel the Menu: **Represent** -> **Represent the selected element with a Single Concept of type Proposition**. A representation list editor is shown (Cf. 4.4). The appropriate representation list is then shown. You must fulfill Meta Information. A concept called Proposition is shown (Figure 20).

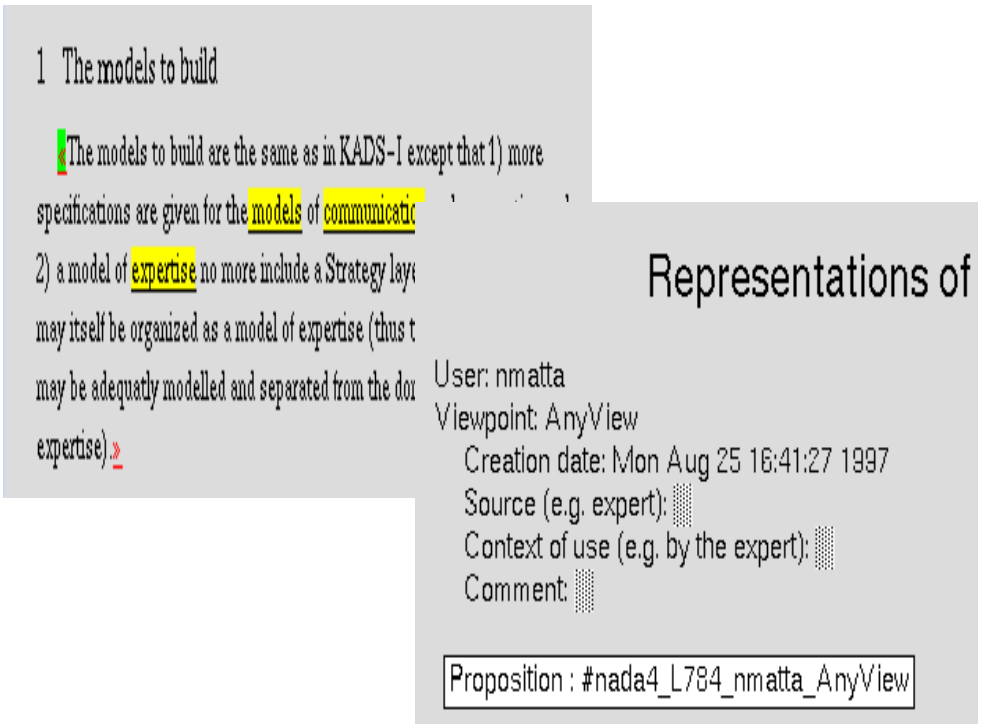


Figure 20 : Represent a CGgraph

✎ Select the referent of the concept

✎ and then click on the right button to show an appropriate Insert Menu. The option: **Generated Referent after individual** allows to insert a Graph.

✎ Another menu is then shown. It allows to choose between: **CGgraph** or **Type Definition**. Choose **CGgraph**.

✎ A menu is then shown. It allows to choose the nature of objects to insert in a graph: **Concept**, **ConceptInclusion** and **ConceptOrRelation**. To insert a concept in a graph, we recommend to show the corresponding concept (for example, by clicking on the corresponding element in the document). Then, select **ConceptInclusion** and click on the concept to include (Cf. 5.3.1).

Beware: If no concept is yet defined in the graph, the first ConceptInclusion will not work. You must insert a concept by choosing **Concept** and then, include the corresponding concept, using **ConceptInclusion** as indicated before. Finally, select the first concept (inserted) and cut it (Option **Cut** in **Edit** Menu). Then, **ConceptInclusion** will work normally, to add other concepts (Figure 21).

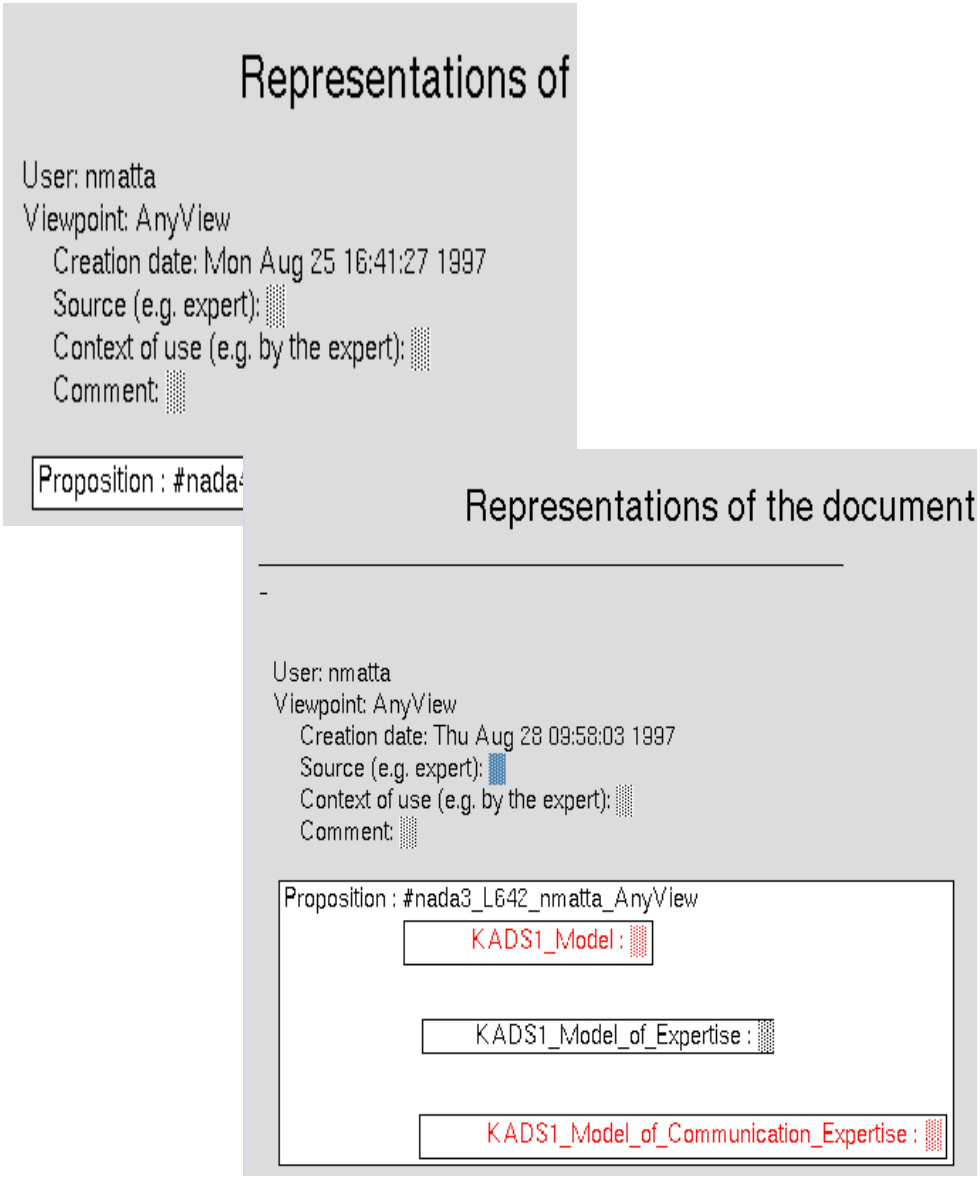


Figure 21 : Include Concepts in the CGgraph

To insert a relation between two concepts already included in the graph:

- select (click with left mouse button on left button of the corner) a concept,
- choose the option **ConceptOrRelation** and then the Option **Relation**

✍ and finally, click (left button) on the name of the first concept and then on the name of the second concept. A draw editor is shown that allows to choose arrows format (Cf. 5.3.2).

To change the name of the relation defined.:

✍ Select the relation and choose to access to the **Access to the Relation Hierarchy** in the **Access/Edit** Menu of the CGKAT panel (Cf. 4.1).

✍ An editor is shown, it presents the Relation Hierarchy. You can enter an approximate name of the concept (in the appropriate field at right button of the hierarchy editor) by clicking three times in this field and then you can enter the name (for more detail, (Cf. 5.4.2). The signature of the relation to be chosen must be coherent with corresponding concepts, it connects, in the graph.

✍ To choose an adequate one, the Option **List relation with their signature** under the button **Other Kinds of relation type lists**, shows all relations defined with their signature.

✍ So, click on the right one

✍ and add it in the graph by pressing the button **Add a relation, or update the selected relation using the below selected type** (Figure 22).

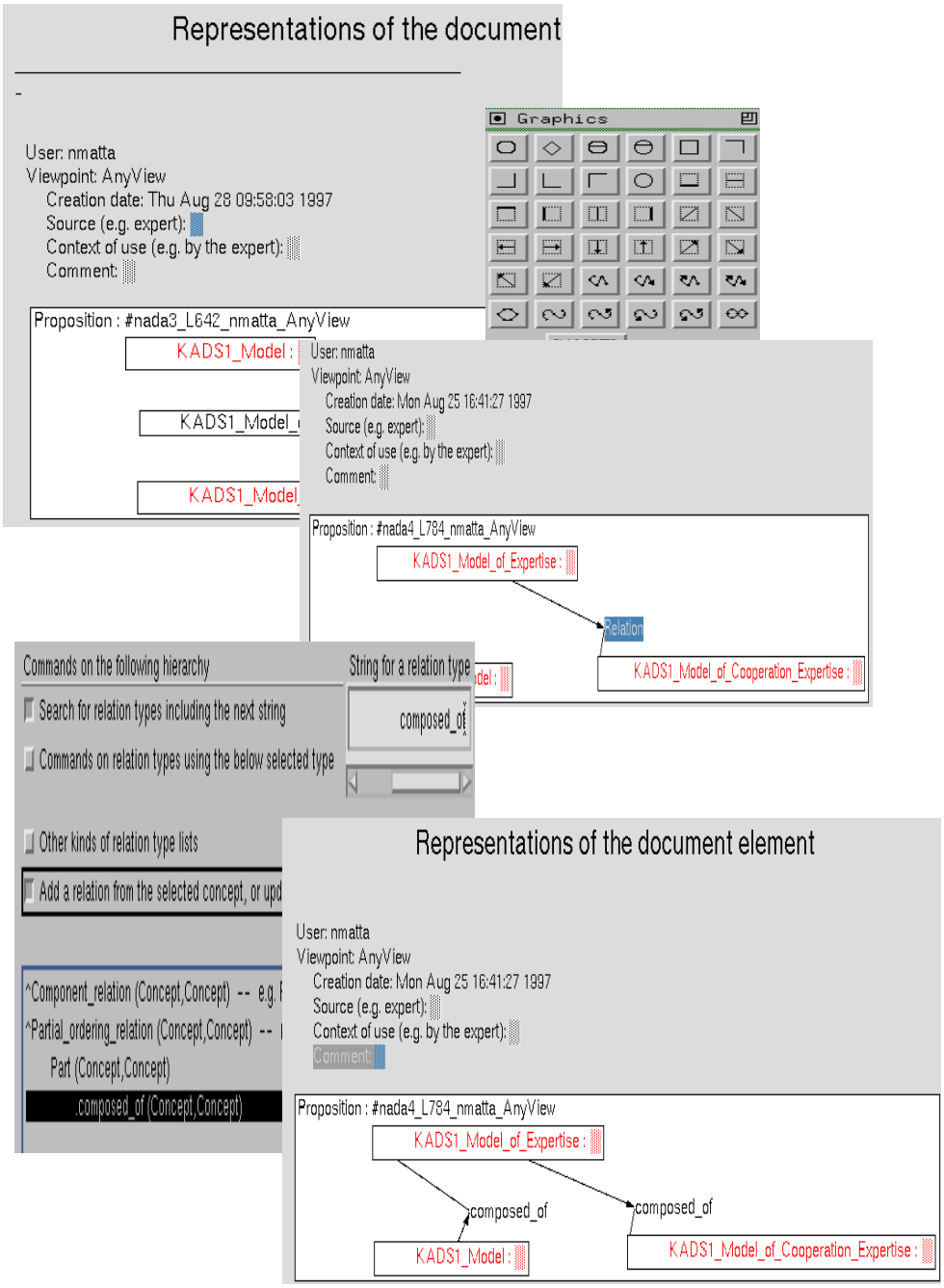


Figure 22 : Define relations between concepts

9.2.3 Represent Type Definition Graph

You can also represent a document element with a Type definition graph. To do this:

- select first the document element to represent,
- then choose in the CGKAT main panel the Menu: **Represent** -> **Represent the selected element with a CG (or a type definition).**
- The appropriate representation list is then shown (Cf. 4.4). You must fulfill Meta information.
- In the bottom of Meta Information, a small rectangle (like a cursor) shows the place to insert a Graph or a type definition. Select this rectangle and then click on the right button to show an appropriate Insert Menu.
- The Option: **Generated CG after (or before) elem** allows to insert a type definition.
- Another menu is shown. It allows to choose between: **CGgraph** or **Type Definition**. Choose **Type Definition** (Figure 23)**.**

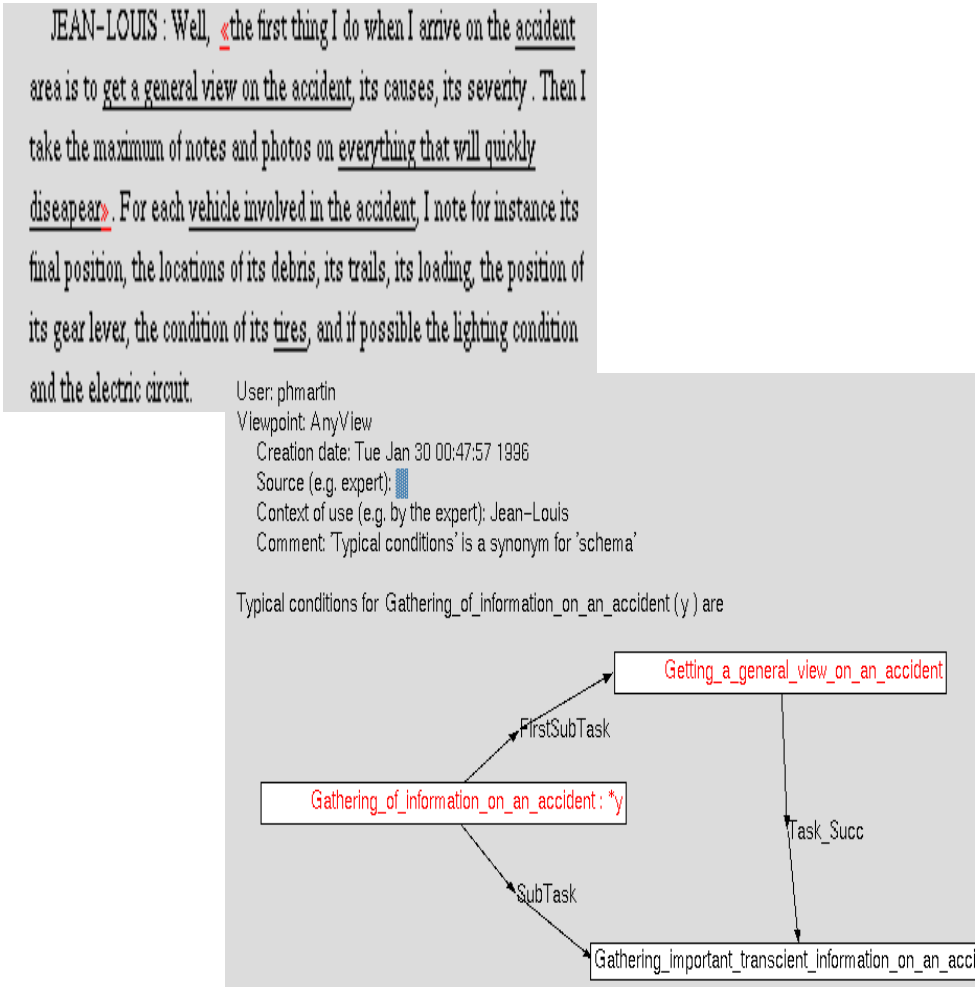


Figure 23 : Represent a type definition graph

✍ A menu is then shown. It allows to choose the nature of objects to insert in a graph: **Concept**, **ConceptInclusion** and **ConceptOrRelation**. To insert a concept in a graph, we recommend to show the corresponding concept (for example, by clicking on the corresponding element in the document).

✍ Then, select **ConceptInclusion** and click on the concept to include (Cf. 5.3.1).

⚠ **Beware**: If no concept is yet defined in the graph, the first ConceptInclusion will not work. You must insert a concept by choosing **Concept** and then, include the corresponding con-

cept, using **ConceptInclusion** as indicated before. Finally, select the first concept (inserted) and cut it (Option **Cut** in **Edit** Menu). Then, **ConceptInclusion** will work normally, to add other concepts (Figure 21).

To insert a relation between two concepts already included in the graph:

- select (click with left mouse button on left button of the corner) a concept,
- choose the option **ConceptOrRelation** and then the Option **Relation**
- and finally, click (left button) on the name of the first concept and then on the name of the second concept. A draw editor is shown that allows to choose arrows format (Cf. 5.3.2).

To change the name of the relation defined:

- Select the relation and choose to access to the **Access to the Relation Hierarchy** in the **Access/Edit** Menu of the CGKAT panel (Cf. 4.1). An editor appears presenting the Relation Hierarchy. You can enter an approximate name of the concept (in the appropriate field at right button of the hierarchy editor) by clicking three times in this field and then you can enter the name (for more detail, (Cf. 5.4.2). The signature of the relation to be chosen must be coherent with the corresponding concepts in the graph (The signature of the relation must correspond to the type of the concepts in the graph).
- To choose an adequate one, the Option **List relation with their signature** under the button **Other Kinds of relation type lists**, shows all relations defined with their signatures.
- So, click on the right one and add it in the graph by pressing the button **Add a relation, or update the selected relation using the below selected type**.

9.3 Search Information

Information search can be performed in two ways: a simple navigation using hypertext-links and a conceptual search using search commands.

9.3.1 Hypertext navigation

Document elements referring to other ones are marked. In fact, elements represented by Single concepts are underlined. Click (Double click left button) on an underlined element and the corresponding single concept is shown. Elements represented by CGgraphs, by Type definitions or by annotations have marks around. Click (left button) on these marks and the corresponding representation is shown. Included concepts and graphs have red as foreground colour. Click (Double click left button) on a graph and the corresponding referred concept or graph is shown. Document elements can be linked using the **Attribute** (**Attribute** Menu) **to Anything**. For example, a graph defined in the document can be linked to a paragraph. Document elements so linked have marks around. Click (Double click left button) on these marks and the corresponding linked element is shown in reverse video.

9.3.2 Conceptual Search

To realize this conceptual search, a number of commands can be used in the interpreter editor or in a shell editor (in this case, you must type ck <command>, i.e. ck spec [cat]). To access to the interpreter editor:

- use **Open Load** Menu in the CGKAT panel (Figure 4)
- and choose **Start a linear interpreter to access or update the KB** option. An editor is shown in which commands can be typed (Cf. 6).

Beware: To enter a command: you must select it (use the first button) and push the **Esc** button or the up arrow (as to go to the father). To type another line in a command, push <ctrl return>. Answers are shown in the same editor.

Examples of commands:

Open documents in which search must be done, using **Open Load** Menu in the CGKAT panel (Figure 4) and **Open Document**.

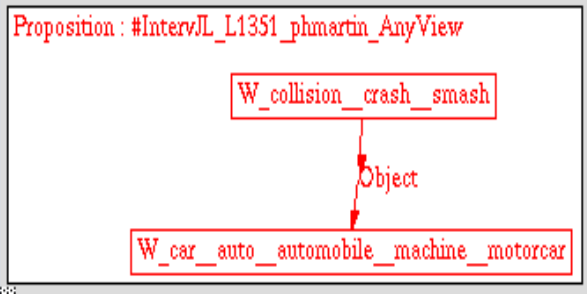
In the Command Interpreter, you can type commands like:

> on CGs
> use CGs

> spec [W_collision__crash__smash]->(Object)->[%car]

Answer:

```
> on CGs
Ok

> use CGs
Ok

> script ex1
Ok

> spec [W_collision__crash__smash]->(Object)->[%car]

--- Generated request: [W_collision__crash__smash]->(Object)->[Cartoon_Character]



--- Generated request: [W_collision__crash__smash]->(Object)->[Cardinality]



--- Generated request: [W_collision__crash__smash]->(Object)->[W_car__auto__automobile__machine
```



> use repr
> spec [W_collision__crash__smash]->(Object)->[%car]

Answer:

```
> use repr
Ok

> script ex1
Ok

> spec [W_collision__crash__smash]->(Object)->[%car]


--- Generated request: [W_collision__crash__smash]->(Object)->[Cartoon_Character]




--- Generated request: [W_collision__crash__smash]->(Object)->[Cardinality]




--- Generated request: [W_collision__crash__smash]->(Object)->[W_car__auto__automobile__machine
```

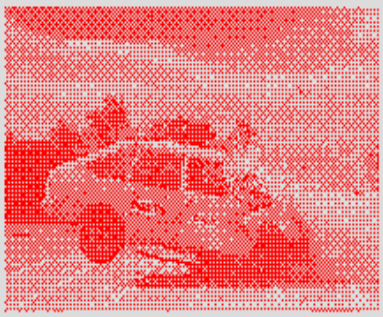
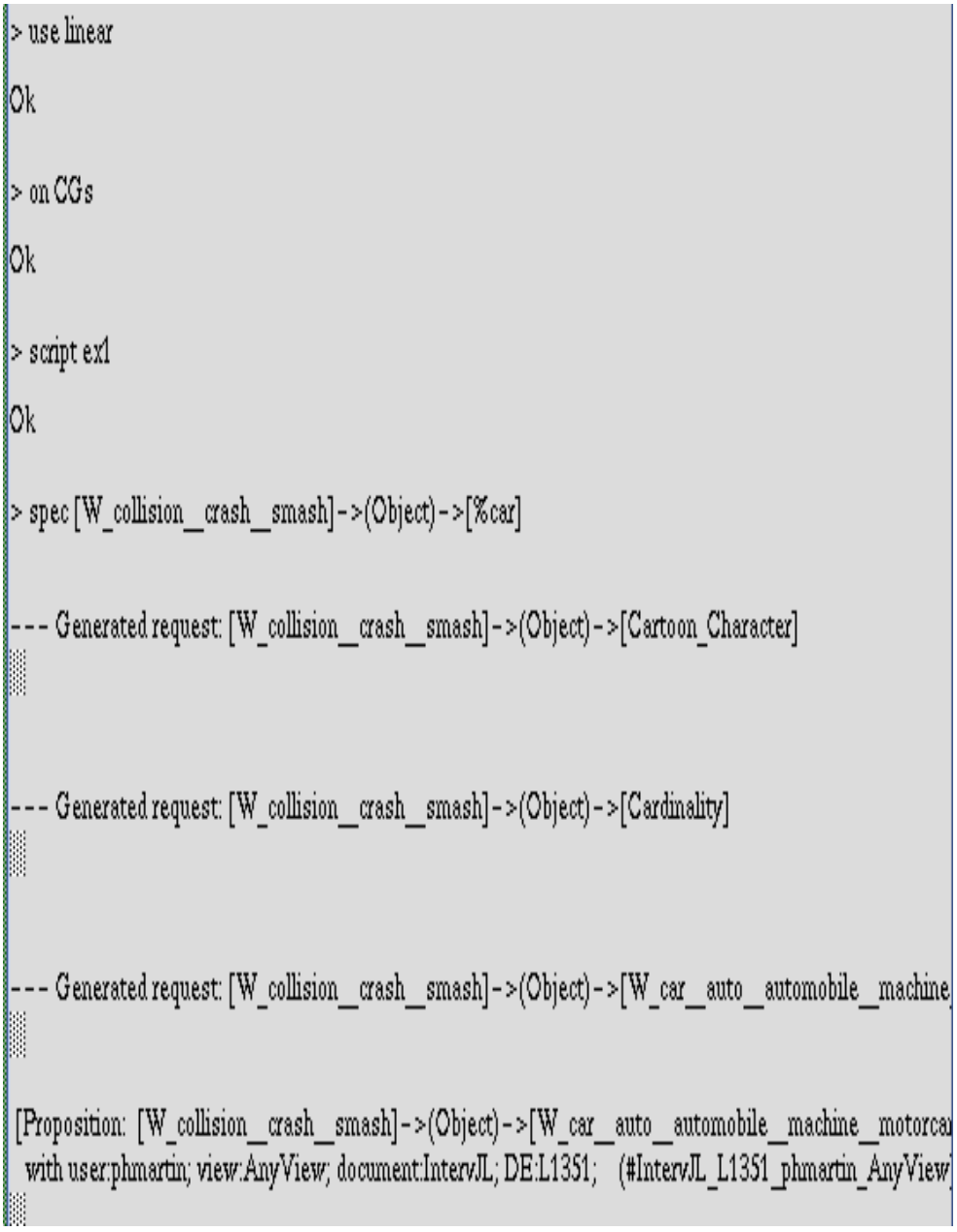


Fig. 0 A car crash (source: INRETS)

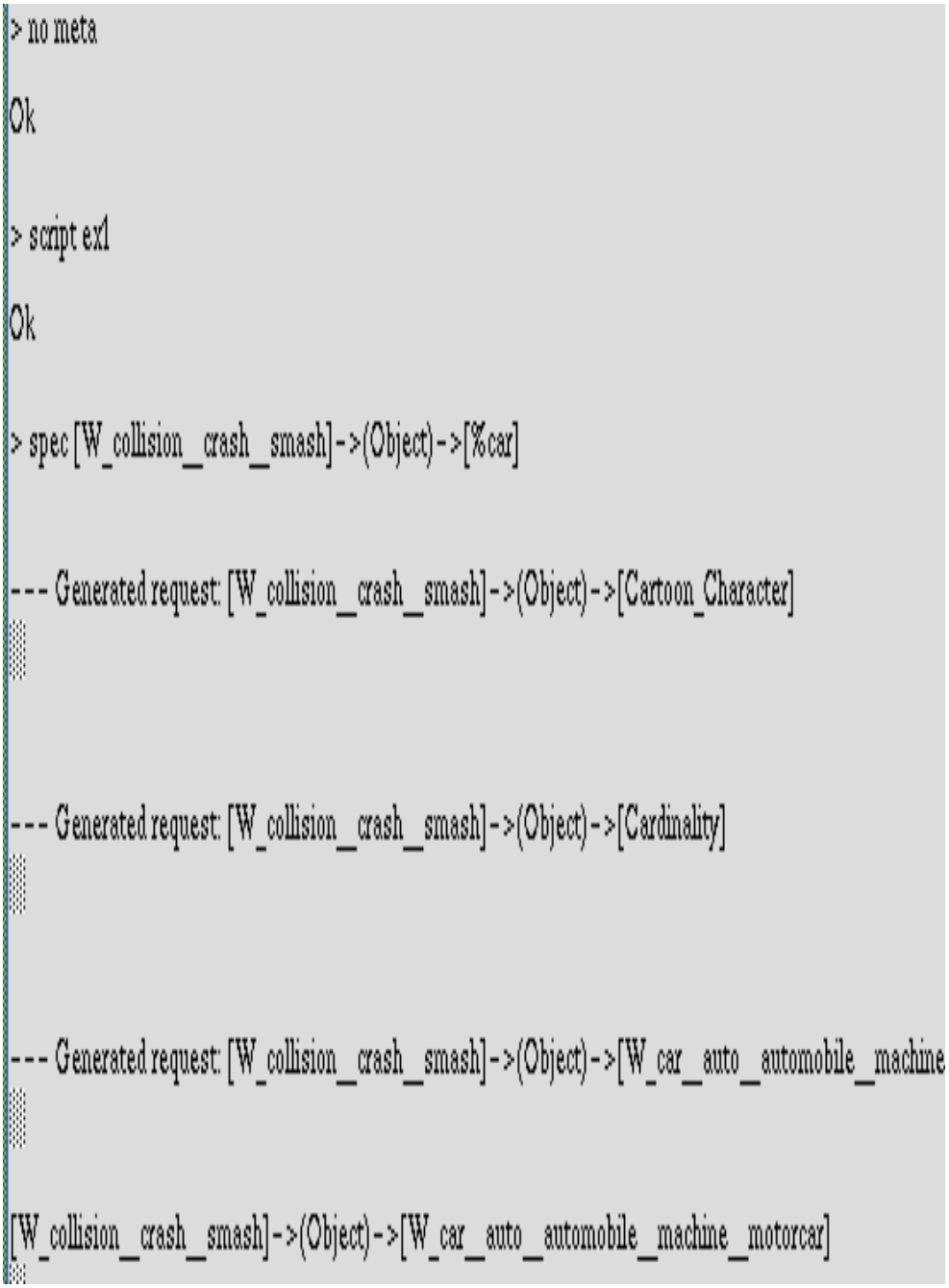
> use linear
> on CGs
> spec [W_collision__crash__smash]->(Object)->[%car]

Answer:



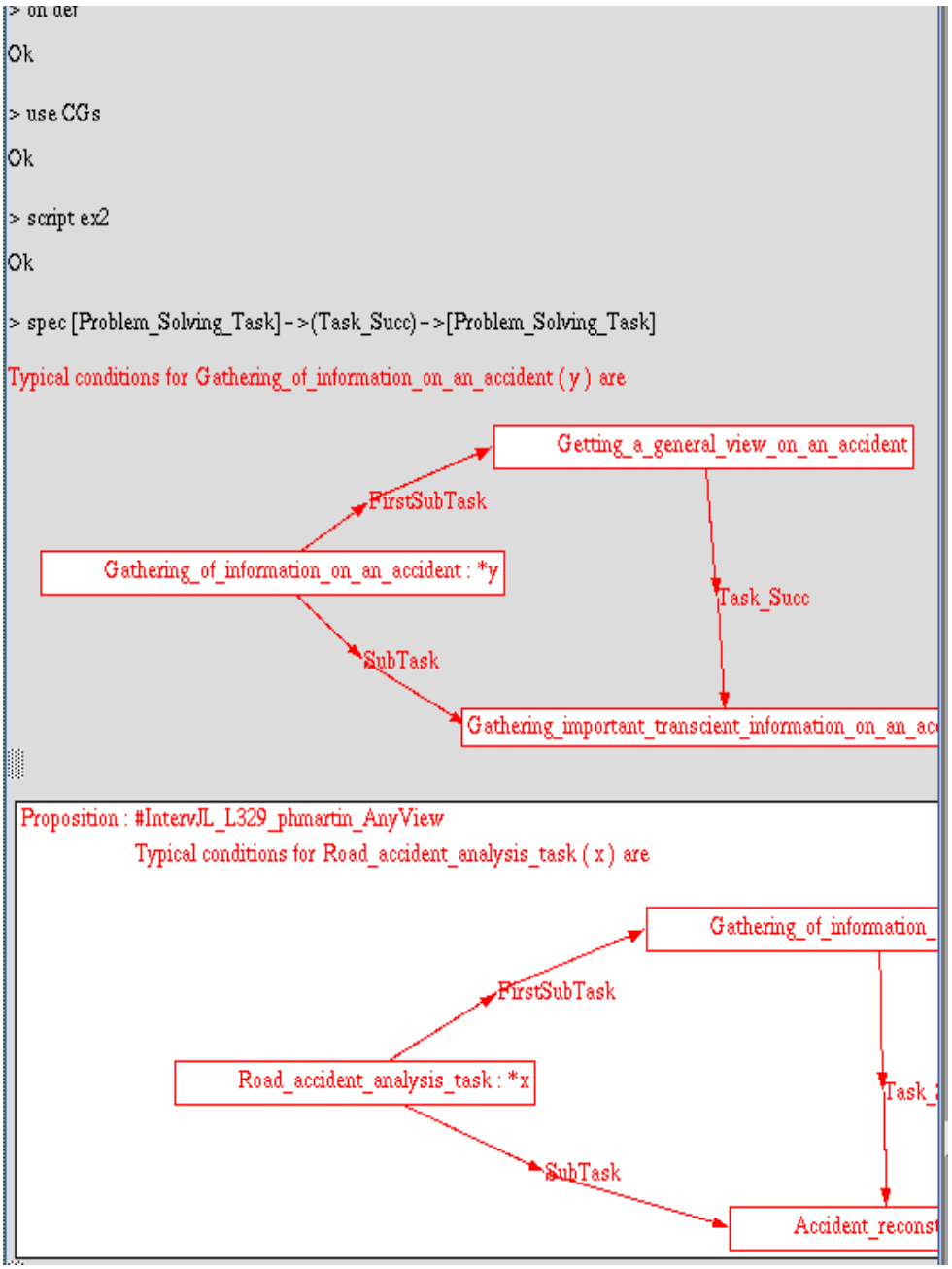
> no meta
> spec [W_collision__crash__smash]->(Object)->[%car]

Answer:



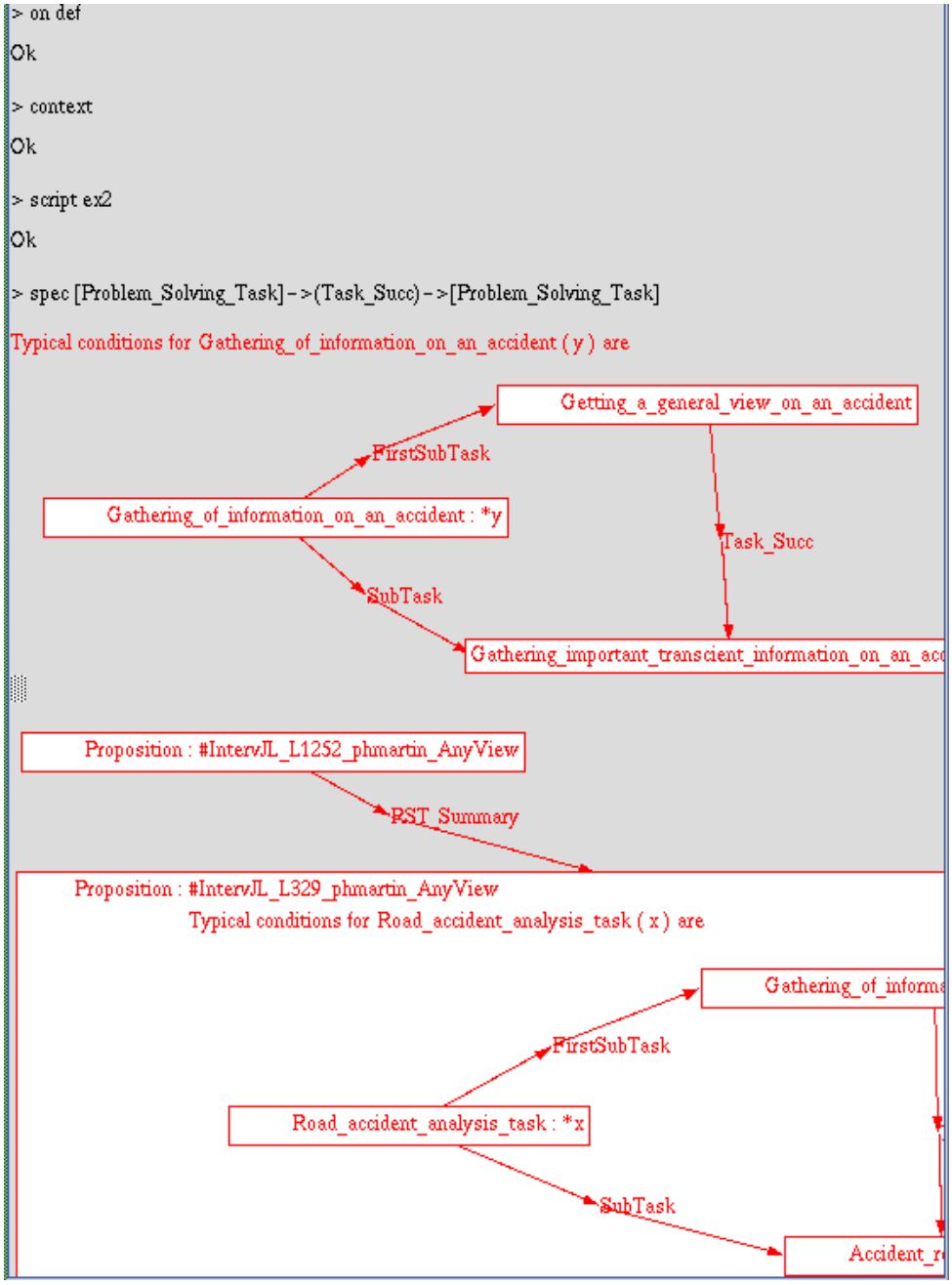
> on def
> use CGs
> spec [Problem_Solving_Task]->(Task_Succ)->[Problem_Solving_Task]

Answer:



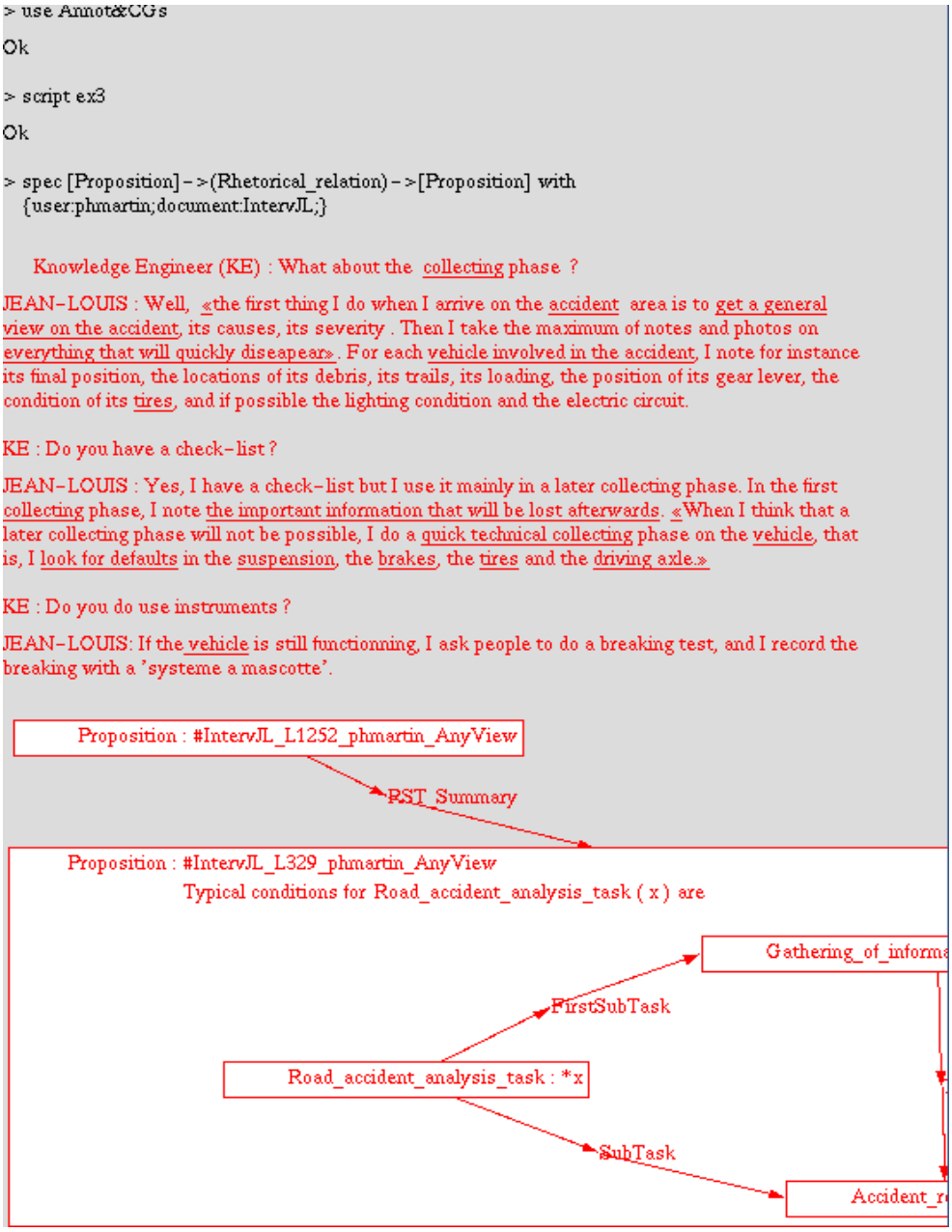
> on def
> context
> spec [Problem_Solving_Task]->(Task_Succ)->[Problem_Solving_Task]

Answer:



> on CGs
> use Annot&CGs
> spec [Proposition]->(Rhetorical_relation)->[Proposition] with {user:phmartin;document:IntervJL;}

Answer:



>spec [Problem_Solving_Task]->(Task_Succ)->[Problem_Solving_Task] with {user:phmartin;document:IntervJL;}

Answer:

```
> script ex4

Ok

> spec [Problem_Solving_Task]->(Task_Succ)->[Problem_Solving_Task] with
  {user:phmartin;document:IntervJL;}

No
```

References

References

[Haemmerlé,95] O. Haemmerlé, CoGITo: une plateforme de développement de logiciels sur les graphes conceptuels. PhD reports, University of Montpellier II, January 1995.

[Miller et al,90] G.A. Miller, C. Felbaum, D. Gross, K. Miller, Five papers on WordNet, CSL Report 43, Cognitive Science Laboratory, Princetown University, July 1990.

[Quint & Vatton,92] V. Quint, I. Vatton, Combining Hypertext and structured documents in Grif, In ECHT 1992.

[Sowa, 84] J. F. Sowa. Conceptual Structures, Information Processing in Mind and Machine. Reading, Addison-Wesley, 1984.

Glossary

Unité de recherche INRIA Lorraine, technopôle de Nancy-Brabois, 615 rue du jardin botanique, BP 101, 54600 VILLERS-LÈS-NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, domaine de Voluceau, Rocquencourt, BP 105, LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
Inria, Domaine de Voluceau, Rocquencourt, BP 105 LE CHESNAY Cedex (France)

ISSN 0249-6399